IMPLEMENTATION OF PUBLIC KEY ELLIPTIC CURVE CRYPTOSYSTEMS OVER $GF(2^n)$

by

Manish Mathur

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

MARCH 1998

IMPLEMENTATION OF PUBLIC KEY ELLIPTIC CURVE CRYPTOSYSTEMS OVER $GF(2^n)$

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Manish Mathur

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
INDIA

March 1998

16 03 48

CERTIFICATE

This is to certify that the present M Tech Thesis work titled **Implementation of Public Key Elliptic Curve Cryptosystems over $GF(2^n)$** , has been carried out by **Manish Mathur** under our supervision and it has not been submitted elsewhere for a degree

M U Siddiqi
Professor
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

S Umesh
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

March, 1998

This work is dedicated to

My Parents

&

God

Acknowledgements

I express my sincere gratitude to my thesis supervisor Dr M U Siddiqi First of all I thank him for introducing me to the fascinating area of **Cryptography** It was his patience and tolerance, and the faith he had in me that helped me to complete the work without any pressure I thank him for providing me the various computing resources and facilities available in his lab and allowing me to work there any time with a great liberty

I thank Dr S Umesh as my co guide whose presence in the lab gave me a lot of sincerity to work in a friendly atmosphere

I thank my teachers Dr S K Bose, Dr D Manjunath Dr R K Bansal, Dr V Sinha and Dr P Sircar I thank Dr Alfred Menezes (*Univ of Auburn*) for their suggestions and comments Thanx to Internet facility available in the lab which tooks me close to him

Special thanks to the following special people for their tuitions Pankaj Bansal, G D Tripathi, Vinay Johar, C Nemade, and R R pandey I thank Pushkar Patwardhan and mangesh Belkhode for a great company in the lab specially during night Words of appreciation are not enough for Shafi who helped me in typing this thesis My heartfelt thanx to Neeraj Gupta Amit Gupta, Manish Gupta, Vivek Sharma, Parag Badge and all Hall5ites who made my stay a memorable one

Abstract

In some applications of public key cryptography it is desirable and perhaps even necessary that key size be as small as possible. Moreover the cryptosystem just needs to be secure enough so that breaking it is not computationally feasible. Most of the known public key cryptosystems are totally insecure if the key size is restricted to about 100-150 bits. Recently Lenstra demonstrated the feasibility to factorize a 450-bit composite integer and La Macchia and Odlyzko computed logarithms in the field over 192-bit prime, while Gordon and McCurley were able to compute logarithms in $F_{2^{401}}$. These results justify the unsuitability of RSA and ElGamal schemes for the applications requiring smaller key sizes. A suitable candidate for such applications that remains is an elliptic curve cryptosystem that provide equivalent security as RSA and other systems but with a much smaller key length. The purpose of this thesis is to provide a practical implementation of these systems.

With the increasing demand of smart card based applications, efficient software implementation of elliptic curve cryptosystems poses a challenge for the cryptographers and software professionals. In this thesis, we have made an attempt to implement them on Pentium and as well as on TMS320C40 digital signal processor using optimizing C cross compiler. The algorithm we adopted is the elliptic curve based ELGamal scheme over Galois field $GF(2^n)$. To obtain minimal complexity in computations, we used optimal normal bases for field arithmetic.

Contents

Chapter 1

Introduction

Data security is an important issue in this present age of computer networking where several types of network attacks, electronic frauds, network hacking are obvious things to happen With the rapidly growing world of communication through computers over an publicly available and highly insecure open medium of **Internet**, there is a heightened demand of the need to protect data during its storage as well as during its transmission from any type of possible attack So computer security and network internetwork security are the two fundamental requirements in the field of information technology where data protection is a crucial issue Now, how this data security is achieved Here an art of secret writing comes into picture, known as **cryptography**

1 1 Cryptography

The word cryptography contains two Greek words, Kryptos, and Graphien The first word means hidden and the second means to write In this way, the art of writing the messages in such a way that they hide their originality is the cryptography But in the world of computer communication and information technology, this art becomes a technology for sending secret information over insecure communication channels such that only intended recipient can read the message

The field of cryptography is thousands of years old Earlier, the scope of this field was considered to be limited to military and diplomatic communities Today, as communication networks are being extensively used by banks, industrial and goverment organization to convey highly sensitive and privileged information information security has become extremely important and much attention has been focused onto this area

The three most needed operations for achieving data secrecy and data authenticity are **encryption decryption** and **digital signature** Encryption is the process of encoding the data at the transmitting end so as to hide its substance The data to be encrypted is called **plaintext** and the encrypted data is called **ciphertext** The process to recover the plaintext from the ciphertext at the receiving end is called decryption Digital signatures, electronic analog of handwritten signatures, are used for proving sender's identity to receiver This process is called authentication These operations are controlled by a cryptographic key or a pair of keys, depending upon the type of cryptosystems used, i e **private key cryptosystems** or **public key cryptosystems**

1 2 Private Key Cryptography

In this type of cryptography the security of the whole cryptosystem relies on a single key This key, a string of bits, is kept secret because disclosure of this key breaks down the security of the whole system To use such a system, sender and receiver initially agree upon a secret key and both possess this key in advance before starting communication They may do this, for example, by physically meeting or by using the services of a trusted courier Now sender encrypts the plaintext by using this secret key and at the receiving end, receiver decrypts the ciphertext using the same key and recovers the plaintext Since, any third person does not have this key so data secrecy is sustained The most widely used private key cryptosystem today is the **Data Encryption Standard** [NBS77] To know more about private key cryptography, see [Sta95 Sch93, Sim91, Denn]

Although private key cryptography is adequate for many applications, it suffers from some problems First is key distribution problem, as a secret channel for selecting a common key may not be available Second is key management problem as every pair of users must share a different secret key so if the number of users is large then the number of keys becomes unmanageable Moreover, no signature possible , as sender and receiver have the same capabilities for encryption and decryption so receiver cannot convince a third party that a message he received from sender in fact originated from sender

To avoid these three deficiencies in private key cryptography, a need was felt to devise some suitable alternative Consequently, in 1976, W Diffie and M Hellman [DH76, DH79] invented public key cryptography

1 3 Public Key Cryptography

The invention of public key cryptography, in 1976 came a new revolution in the field of cryptography and today, where several good public key cryptosystems have already come in the market, the research in this area is continuously going on to devise better and better systems The public key cryptosystems are the two key cryptosystems, wherein each user has both a public and private key and the two users can communicate knowing only each other s public keys The public keys of all the users are publicly available in any database over the network so, any user can get the public key of any other user from this database Now if a user wants to send some message to any other user, the sender encrypts the message by using his/her private key and the receiver's public key At the receiving end receiver decrypts the ciphertext by using his/her private key and the sender s public key Since private key of the receiver does not possess by any other, data secrecy is sustained because no other person can decrypt the received data Moreover, data authenticity is also achieved since no other person can encrypt the data using transmitter's private key In this way, the three deficiencies of private key cryptography are automatically removed in this type of cryptography

Security of the public key cryptosystems relies on a very useful and inter esting property of some special type of mathematical functions known as one way functions A one way function is an invertible function which is very easy to compute in the one direction but very difficult in the reverse direction One example is it is easy to compute n = pq, where p and q are two large primes but it is very difficult to find p and q if their product, n is known This property of one way function is well known as **integer factorization problem** The other example is it is easy to compute $b = a^x$, if a and x are given but it is very difficult to find x from this equation under a large finite multiplicative group This one way property is well known as **discrete logarithm problem** This problem is considered to be very difficult if the order of the group G is very large The points on an elliptic curve also form a one way function as we will see in chapter 2

1 4 RSA Today's most widely used public key cryptosystem

The **RSA** cryptosystem was invented in 1977 by Rivest Shamir and Adleman [RSA78] and today it is the most widely used public key cryptosystem The security of RSA system relies on the factorization of a large integer To set up this system each user A picks two large primes p and q and computes their product n = pq A's public key is the pair of integers (n,e) and his/her private key is d The arithmetic is done over a finite multiplicative group of units in the integers modulo n, as shown below

Encryption $C = M^e$ mod n, where M is the message block ∈ [0, n-1]
Decryption $M = C^d$ mod n, simultaneously e and d satisfy the relation
ed mod $\Phi(n) = 1$, where $\Phi(n) = (p\ 1)(q\ 1)$

From the above arithmetic, it is clear that the security of the RSA cryptosystem relies on the factorization of n In other words, breaking the RSA is equivalent to factoring n A great deal of progress has been made in devising efficient algorithms for factoring integers and thus threatened the security of RSA With the current state of our knowledge and technology if p and q are each about 100 decimal digits then n is 200 digits, then factoring n is an intractable problem So, if n d, and e are each 200 decimal digits(664 bits), the storage requirement per user are about 2,000 bits Since both e and n must be made public, the public storage requirements are thus about 1 3 kilobits per user Simultaneously, to encrypt or decrypt a 664 bit number requires 1 2 multiplications in modular arithmetic per bit, or about 1,000 multiplications total In this regard of storage and processing requirement, **RSA** is **unsuitable for applications where processing power and storage space are limited** One such application is **smart card** where processing power and storage space are crucial issues

1 5 Cryptographic Smart Card an ultimate secure device

Internet commerce, pay channel television, public telephones, data access control etc are such areas that have led to heightened demand of an information security device, known as smart card This pocket size plastic card looks like an ordinary credit card but contains memory and processing capabilities The smart card, in fact is a multipurpose, tamper resistant security device which is equipped with volatile and non volatile memory and a microprocessor, all on a single 20 mm^2 chip, for carrying out the computations for various security services like encryption, decryption and digital signature The basic advantage of the smart card is that the secret key of the user is stored in a nonvolatile memory and never leaves the card All the processing required for encryption and decryption is done on the card itself

Since, development of an efficient smart card cryptosystem is an important issue today, there is a need to point out the limitation of current system and to go for some better system to avoid these limitations Due to its small size and small processing power, the selected public key algorithm for smart card should have storage and processing requirements as minimum as possible Presently, the

RSA is being widely used for smart cards and significant advances have also been made on efficient implementation of these cryptosystems including, custom VLSI chips [OSA92 VVDJ92, Bri89] and very efficient digital signal processor software implementations on Texas Instruments TMS32010 [PB86] and Motorola DSP56000 [DJ91] Keeping in mind the present state of computational technology, the RSA system requires modular arithmetic of at least 512 bit integers The chip [IWSD92] designed to do modular multiplication of 512 bit numbers has about 50,000 gates while the chip designed to perform arithmetic in the field $F_{2^{593}}$ has about 90 000 gates With current technology placing these devices on a 20 mm^2 smart card chip is a complicated and expensive procedure Moreover, under the recent improvement in integer factorization and parallel processing, the security of these systems is facing a serious threat Recently, Lenstra et al were able to factor a 450 bit composite integer using distributed processing Hence for the cryptosystems to be secure the size of the modulo integers needs to be increased further This leads to further increase the VLSI implementation related problems

Because of all these reasons, alternatives of RSA are being looked with great interest One suitable alternative, that may become a proper substitute for RSA in future is **elliptic curve cryptosystems**

1 6 Elliptic Curve Public Key Cryptosystems a proper substitution of RSA in future

The theory of elliptic curves is not a very new in the field of algebraic geometry and number theory but application of these curves in the field of cryptography is a new idea Elliptic curves were first suggested in 1985 by N Koblitz [KOB87] and V Miller [MILL85] for implementing public key cryptosystems The points on an elliptic curve over a finite field form an abelian group The binary operation for this abelian group involves few arithmetic operations in the field over which curve is defined Moreover, the discrete logarithm problem in this group is very much difficult as compared to the discrete logarithm problem over the field itself of the same sıze Although there are several algorıthms for findıng logarıthm ın a finıte field but there are only a few whıch are applıcable over any arbıtrary group In partıcular there seems to be no proper choıce of any algorıthm for the group formed b͏y the poınt on the curve Due to thıs reason, the field sıze over whıch curve ıs defined can be made smaller, for example $F_{2^{155}}$ or even $F_{2^{135}}$, wıthout compromısıng on securıty Furthermore, over a gıven field, there ıs a number of dıfferent ellıptıc curves possıble Consequently usıng the same hardware, user can change the chosen curve perıodıcally for gaınıng extra securıty

Sınce, ellıptıc curve publıc key cryptosystems provıde equıvalent securıty as the RSA even wıth shorter key lengths and hence provıde much smaller bandwıdth, memory and processıng requırements, these systems may be very useful ın the design of smart card based cryptosystems From the poınt of vıew of the hardware ımplementatıon, a VLSI chıp $F_{2^{155}}$ ASIC (applıcatıon specıfic ıntegrated cırcuıt) has been buılt [AMV93] to demonstrate the feasıbılıty of such devıces It has only about 11 000 gates and the complete ellıptıc curve cryptosystem over $F_{2^{155}}$ could be fabrıcated and use up less than 4% of the $20mm^2$ desıgnated for a smart card processor

1 7 What has been done ın thıs thesıs

Sınce ellıptıc curve cryptosystems may offer equıvalent securıty as RSA, even wıth much smaller memory and processıng requırements, a lot of work ın thıs partıcular area of cryptography ıs beıng done wıth the ıncreasıng popularıty of smart card and electronıc commerce Thıs thesıs also ıs aımed on the software ımplementatıon of ellıptıc curve publıc key cryptosystems Sımultaneously, desıgn ıssues of smart card have been dıscussed A brıef survey on electronıc commerce ıs also ıncluded

The work, done ın thıs thesıs, ıs as follows

- Subroutınes for efficıent arıthmetıc ın $GF(2^n)$ are wrıtten
- A package for encryptıon and decryptıon wıth very small key lengths of about 100 bıts For example, over $GF(2^{113})$

This package may be useful in applications where the public keys are desired to be as small as possible without loosing any security. One such application is secure e-mail exchange or secure transmission of message by fax. This has been successfully tested on the Pentium 100 MHZ/DOS and Pentium 150 MHZ/Linux.

- A package for encryption and decryption over much higher fields upto $GF(2^{400})$, on TMS320C40 25 MHZ digital signal processor using optimizing C cross compiler.

- A brief study on smart card and e-commerce.

All the implementation work has been done in today's most general purpose language ANSI C, without using any ready made package. So, this must be useful one at application level where one would not like to install whole package for getting services like email security.

1.8 Organization of Thesis

The thesis is organized into 6 chapters, including the present one. Chapter 2 introduces the notion of elliptic curves over $GF(2^n)$ and covers some arithmetic results which are necessary for implementation. Chapter 3 describes the elliptic curve discrete logarithm problem with its security aspects. Various algorithms have been discussed to compute it. It also suggests that how an elliptic curve should be chosen for building a cryptosystem so that all algorithm fail to compute DLP. In chapter 4, we discuss some elliptic curve public key cryptosystems. An elliptic curve based smart card cryptosystem has also been designed in this chapter. In chapter 5, we have given the implementation work in detail. All algorithms have been clearly specified with their implementation results. Chapter 6 concludes the thesis. The thesis also has one appendix in which a brief introduction on electronic commerce has been given.

Chapter 2

An Introduction to Elliptic Curves

As we mentioned earlier, the points on an elliptic curve over a finite field constitute an abelian group In this chapter, we will introduce the notion of elliptic curves and see the arithmetic of these points over this abelian group and collect some results, which have been used in the implementation of elliptic curve public key cryptosystems

Note Before reading this chapter, the readers should be familiar about algebraic number theory, class field theory and algebraic geometry For this, they may refer [PS92, Ono90, Cha88, IR82, Hec93, Coh78, Ros94, Sil94, Fra95, BJN94]

2.1 Weierstrass Equation

Let K is a finite field, F_q, containing q elements, where q is a prime power Let $\bar{K}$ denotes its algebraic closure and $P^2(\bar{K})$ denotes the projective plane over K A Weierstrass equation is a homogeneous (projective) equation of degree 3 of the form

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3,$$

where $a_1, a_2, a_3, a_4, a_6 \in \bar{K}$

If for all points $P \in P^2(\bar{K})$ satisfying

$$E_1 \quad y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$
$$E_2 \quad y^2 + \bar{a}_1xy + \bar{a}_3y = x^3 + \bar{a}_2x^2 + \bar{a}_4x + \bar{a}_6$$

are isomorphic over K, denoted $E_1/K \cong E_2/K$, if and only if there exists $u, r, s, t \in K, u \neq 0$, such that the change of variables

$$(x, y) \longrightarrow (u^2x + r, u^3y + u^2sx + t)$$

transforms equation E_1 to equation E_2 The relationship of isomorphism is an equivalence relation

2.3 Group Law

As said earlier, the points on an elliptic curve constitute an abelian group Now we will see the addition rules over this group Let E be an elliptic curve given by the non-homogeneous Weierstrass equation The addition rules are given below

For all $P, Q \in E$

1 $\mathcal{O}$ $+P = P$ and $P + \mathcal{O} = P$ So $\mathcal{O}$ serves as the identity element

2 $-\mathcal{O} = \mathcal{O}$

3 If $P = (x_1, y_1) \neq \mathcal{O}$, then $-P = (x_1, -y_1 - a_1x_1 - a_3)$ For a given elliptic curve P and $-P$ are the only points with x-coordinate equal to x_1

4 If $Q = -P$, then $P + Q = \mathcal{O}$

5 If $P \neq \mathcal{O}$, $Q \neq \mathcal{O}$, $Q \neq -P$, then let R be the third point of intersection of either the line $\bar{PQ}$ if $P \neq Q$, or the tangent line to the curve at P if $P = Q$, with the curve Then $P + Q = -R$ Or in other words, $P + Q + R = \mathcal{O}$ (from the axiom fourth)

2.4 Addition of Two Points on an Elliptic Curve

In this section we will describe the axiom fifth of the group law in some detail Let E be an elliptic curve over a finite field GF(q) together with the point at infinity $\mathcal{O}$ It is obvious from the degree of the Weierstrass equation that any line in affine plane will intersect the curve at exactly three points, say P, Q, R In case of the tangent line, P, Q, R may not be distinct

Let $P, Q \in$ E, L the line passing through P and Q (tangent if $P = Q$), and R the third point of intersection of L with E Let L' be the line connecting R and $\mathcal{O}$ as shown below Then $P + Q$ is the point such that L' intersects E at $R, \mathcal{O}$ and $P + Q$ This can be easily verified that $P + Q = -R$ and hence, $P + Q + R = \mathcal{O}$ from the fourth axiom of the group law

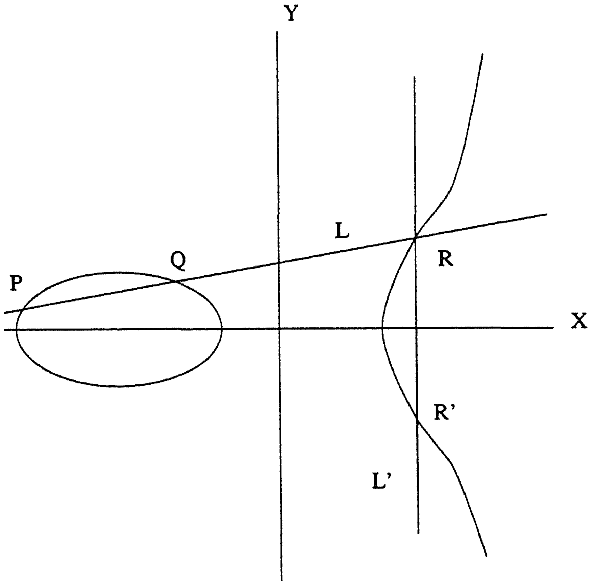

Figure 2 1 Addition of two points over an Elliptic Curve

Hence we see that the addition of two points on an elliptic curve is the inverse of the point at which the line passing through those two points cuts the curve Now we will see the explicit rational formulae for the coordinates of the resultant point $P + Q$ in terms of the coordinates of P and Q

Let $P = (x_1, y_1), Q = (x_2, y_2), P+Q = (x_3, y_3)$ Then the slope of L, joining

P and Q, is

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q, \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{if } P = Q \end{cases}$$

If $c = y_1 - mx_1$, then the equation defining L is $y = mx + c$ Now, by substituting $y = mx + c$ into Weierstrass equation and doing some simple algebraic manipulation, the coordinates of third point of intersection of line L with the curve E can be obtained and given by

$$x_3 = m^2 + a_1m - a_2 - x_1 - x_2$$
$$y_3 = -(m + a_1)x_3 - c - a_3$$

So, if $P, Q \in$ E/K, then computing $P + Q$ involves just a few arithmetic operations in the field K Hence if K is a finite field, then computing $P + Q$ takes polynomial time

2.5 The Discriminant and j-invariant

In this section, we will introduce two useful quantities for a given elliptic curve Let E be a curve given by a non-homogeneous Weierstrass equation Define the quantities

$$d_2 = a_1^2 + 4a_2$$
$$d_4 = 2a_4 + a_1a_3$$
$$d_6 = a_3^2 + 4a_6$$
$$d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$$
$$c_4 = d_2^2 - 24d_4$$
$$\Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6$$
$$j(E) = c_4^3/\Delta$$

The quantity Δ is called the *discriminant* of the equation and $j(E)$ is called the $j - invariant$ of E if $\Delta \neq 0$ These two quantities play a significant role in defining the non-singularity and isomorphic property of elliptic curves A given elliptic curve E is non-singular if and only if $\Delta \neq 0$ From isomorphism viewpoint, two elliptic curves E_1/K and E_2/K are isomorphic over K, if $j(E_1) = j(E_2)$

2.6 Curves over K, char(k) = 2

Since we have used the curves over $GF(2^n)$ to implement the cryptosystems, we will consider only the curves over K with char(K) = 2

Let K be a field of characteristic 2, and let E/K be the elliptic curve given by the Weierstrass equation

$$\text{E} \quad y^2 + \bar{a_1}xy + \bar{a_3}y = x^3 + \bar{a_2}x^2 + \bar{a_4}x + \bar{a_6}$$

For this curve the $j - invariant$ is $(\bar{a_1})^{12}/\Delta$

If $j(E) \neq 0$, then by the property of isomorphism, the admissible change of variables

$$(x,y) \longrightarrow \left(\bar{a_1}^2 x + \tfrac{\bar{a_3}}{\bar{a_1}}, \bar{a_1}^3 y + \tfrac{\bar{a_1}^2\bar{a_4} + \bar{a_3}^2}{\bar{a_1}^3}\right)$$

transforms E to the curve

$$E_1/K \quad y^2 + xy = x^3 + a_2x^2 + a_6$$

For this transformed curve $\Delta = a_6$ and $j - invariant = 1/a_6$

If $j(E) = 0$, then the admissible change of variables

$$(x,y) \longrightarrow (x + \bar{a_2}, y)$$

transforms E to the curve

$E_2/K \quad y^2 + a_3 y = x^3 + a_4 x + a_6$

For this curve, $\Delta = a_3^4$ and j-*invariant* $= 0$

Now we will see the addition formulas for the two given points P and Q over these transformed curves, both for the curves with $j(E) \neq 0$ as well as the the curves with $j(E) = 0$

Addition formula when $j(E) \neq 0$

Let $P = (x_1, y_1) \in E_1$, then $-P = (x_1, y_1 + x_1)$ If $Q = (x_2, y_2) \in E_1$ and $Q \neq -P$, then $P + Q = (x_3, y_3)$, where

$$x_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right)^2 + \frac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a_2, & P \neq Q \\ x_1^2 + \frac{a_6}{x_1^2}, & P = Q \end{cases}$$

and

$$y_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right)(x_1 + x_3) + x_3 + y_1, & P \neq Q \\ x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3, & P = Q \end{cases}$$

Addition Formula when $j(E) = 0$

Let $P = (x_1, y_1) \in E_2$, then $-P = (x_1, y_1 + a_3)$ If $Q = (x_2, y_2) \in E_2$ and $Q \neq -P$, then $P + Q = (x_3, y_3)$, where

$$x_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right)^2 + x_1 + x_2, & P \neq Q \\ \frac{x_1^4+a_4^2}{a_3^2}, & P = Q \end{cases}$$

and

$$y_3 = \begin{cases} \left(\frac{y_1+y_2}{x_1+x_2}\right)(x_1 + x_3) + y_1 + a_3, & P \neq Q \\ \left(\frac{x_1^2+a_4}{a_3}\right)(x_1 + x_3) + y_1 + a_3, & P = Q \end{cases}$$

2.7 Order of an Elliptic Curve

If elliptic curve E is given by the non-homogeneous Weierstrass equation then the number of solutions of this equation together with the point at infinity $\mathcal{O}$ is called the order of the elliptic curve E Let E be defined over F_q Let $q = p^m$, where p (a prime) is the characteristic of F_q We denote the number of points in $E(F_q)$ by $\#E(F_q)$ Since the maximum degree of y in the Weierstrass equation is 2 so, it has at most 2 solutions for each choice of $x \in F_q$ and hence $\#E(F_q) \leq 2q+1$ Remember that, the added one indicates the point at infinity

In 1985, Schoof [Sch85] presented a polynomial time algorithm for computing $\#E(F_q)$ The algorithm has a running time of $O(\log^8 q)$ bit operations, and is rather cumbersome in practice Buchmann and Muller [BM91] combined Schoof's algorithm with Shanks' baby-step giant-step algorithm and were able to compute $\#E(F_q)$ over 27-digit prime field in 4 5 hours on a SUN-1 SPARC-station

Now we will see the bound on the order of an elliptic curve, given by Hasse in his following theorem

2.8 Hasse Theorem

Let $\#E(F_q) = q + 1 - t$ Then $|t| \leq 2\sqrt{q}$

An important consequence of Hasse's Theorem is that we can pick points P uniformly and randomly on an elliptic curve $E(F_q)$ in probabilistic polynomial time

If t is divisible by the characteristic of F_q then the elliptic curve $E(F_q)$ is said to be *supersingular* Otherwise it is called $non - supersingular$ In fact, an elliptic curve is super-singular if and only if $t^2 = 0, q, 2q, 3q$, or $4q$

In 1990, Menezes, Okamoto and Vanstone demonstrated that the discrete logarithm problem on a *supersingular* elliptic curve can be reduced to the discrete logarithm problem in a finite field This result means that we should avoid the set of super-singular curves if we want to have a cryptosystem whose cracking problem is of fully exponential complexity

With this we conclude this chapter Actually, we covered here only an elementary introduction of elliptic curves and gave only some important results For going into details, the interested readers may refer [Men93, Cass91, Hus87, Kob84, Lang78] For an introduction to the general theory of algebraic curves, the readers may refer to [Fult69, More91]

Chapter 3

Elliptic Curve Discrete Logarithm Problem : Attacks and Remedies

The discrete logarithm problem in some finite group G is the foundation of the building of many public key cryptosystems that are being used today Elliptic curve cryptosystems are also designed on the bases of discrete logarithm problem One of the best example of such type of cryptosystems is ElGamal cryptosystem, proposed by T ElGamal in 1985 [ElG85] Since, the security of the whole cryptosystem relies in the presumed intractability of the discrete logarithm problem, it has received a great deal of attention in recent years, and the numerous algorithms have been devised to solve it In this chapter, first we will introduce DLP over a finite abelian group and a group constitute by the points of an elliptic curve over a finite field Then we will consider various algorithms that threaten the security of the DLP based cryptosystems After this we will determine the conditions for which a known algorithm will fail and hence security is sustained Finally, we will see that how the elliptic curve cryptosystems are built by employing the discrete logarithm problem

3.1 Discrete Logarithm problem (DLP)

The discrete logarithm problem (DLP) in a finite multiplicative group G refers to computation of x for two elements a, b of G such that $a^x = b$ The integer x is said to logarithm of b to base a, i e $x = \log_a b$ This problem is known to be very difficult if the group order is large The intractability of this problem motivated Diffie and Hellman to introduce the concept of public key cryptography which exploits the difficulty of the finding discrete algorithm in a finite group as a security measure Initially, Diffie and Hellman's idea was limited to discrete logarithm problem in multiplicative group of $GF(p)$ but in 1985 it was generalized by ElGamal for any finite abelian group The application of DLP in cryptography has been one of the reasons for increased attention towards solving this problem in the field of **cryptanalysis** Consequently, several algorithm have been devised in the recent past for finding logarithms in finite abelian group

Since, the ongoing research in the area of solving DLP seriously threatens the security of most of the existing public key cryptosystems, the search for more difficult DLP has been of great interest for cryptographers The elliptic curve based DLP is one of the outcome of this search The DLP in an elliptic curve group refers to the computation of k for given $P, Q \in E$ such that

$$Q = kP = \overbrace{P + P + \quad + P}^{k\ times}$$

This one way property provide a very difficult DLP even with a very small field size (of the order of $F_{2^{135}}, or F_{2^{155}}$) over which the curves are defined

Now in the next section, we will briefly introduce various algorithms for finding discrete logarithm in a finite abelian group In each case, we will also compare the complexity of elliptic curve group based DLP with that of finite abelian group based DLP

3.2 Various Attacks on DLP Based Cryptosystems

There are various algorithms known for finding discrete logarithms in a finite abelian group and hence attack the security of DLP based cryptosystems The algorithms can be categorized as follows

1 Algorithms which work in arbitrary groups (square root methods)

2 Algorithms which work in arbitrary groups but exploit the subgroup structure (Pohlig-Hellman method)

3 The index calculus methods

4 MOV reduction attack for elliptic curves only

Now we proceed to briefly describe each of these methods

3.2.1 Square Root Methods

These algorithms, work in any arbitrary cyclic group, are called square root method because their computational complexity is of the order of square root of the size of the group Let G denotes a finite abelian group of order m with α as the generator element Let $m' = \lceil\sqrt{m}\rceil$ One of the famous square root methods is Baby-Step Giant Step method

Baby-Step Giant Step method

Let $\alpha^x = \beta$ Hence, the problem is to find $x = \log_\alpha \beta$ The algorithm begins with precomputing a list of pairs (i, α^i) for $0 \le i \le m'$ and storing it in memory Now for each $j, 0 \le j \le m'$, compute $\beta\alpha^{-jm'}$ and check by applying binary search whether it equals any of α^i in the stored list If the match is found for some i and j then

$$\beta\alpha^{-jm'} = \alpha^i$$
$$\Rightarrow \beta = \alpha i + jm'$$
$$\Rightarrow \log_\alpha \beta = i + jm'$$

By using some mathematical manipulation, it is easy to see that this algorithm requires storage of m' entries and $O(m' \log m')$ steps for an arbitrary group In case of elliptic curve group, the storage list will consists of multiples of generator point which further increase the storage requirements

Now, if we select a finite abelian group (any group, not necessarily elliptic curve group) of the order of 10^{30}, then this algorithm will not be feasible Hence, the group to be selected for secure cryptosystems are restricted to have size greater than 10^{30}

3.2.2 Pohlig-Hellman Method

Let G be an arbitrary group of the order m The method proceeds with factorization of m to determine various subgroups of G and then computes the discrete logarithm in each of the subgroup by using square root method Finally, Chinese remainder theorem [IR82] is applied to obtain the required result Let

$$m = \prod_{i=1}^{t} p_i^{e_i}$$

where p_i are primes and e_i are exponents Let $x = \log_\alpha \beta$ The algorithms begins with computation of $z_i = x \; mod \; p^{e_i}$ for each i
suppose that $z_i = \sum_{j=0}^{e_i-1} z_{ij} p_i^j$ where $0 \leq z_{ij} < p_i$
Let γ_i be the p_i^{th} root of unity in G, i e $\gamma_i = \alpha^{m/p_i}$ Then

$$\begin{aligned} \beta^{m/p_i} &= \alpha^{xm/p_i} \\ &= \gamma_i^{\sum_{j=0}^{e_i-1} z_{ij} p_i^j} \\ &= \gamma_i^{z_{i0}} \end{aligned}$$

Now z_{i0} can be computed using Baby-Step Giant-Step method in $O(\sqrt{p_i}(\log p_i))$ steps and with $O(\sqrt{p_i})$ storage requirement For the computation of z_{i1}, we see that

$$\begin{aligned} (\beta\alpha^{-z_{i0}})^{m/p_i^2} &= (\alpha^{\sum_{j=1}^{e_i-1} z_{ij} p_i^j})^{m/p_i^2} \\ &= \gamma_i^{z_{i1}} \end{aligned}$$

In this way z_{ij} can be computed for all values of j Now for all i, z_i can be computed by repeating above procedure After having all z_i, the required logarithm x can be computed using Chinese remainder theorem

From the complexity viewpoint, this algorithm requires $O(\sum_{i=1}^{t} \log p_i)$ storage elements and $O(\sum_{i=1}^{t} e_i(\log m + \sqrt{p_i} \log p_i))$ steps For this attack to be failed, the order of the group G must contain a large prime factor of the order of at least 30 decimal digits integer

3.2.3 Index Calculus Method

Although this attack is the most powerful attack to find the discrete logarithm, it does not apply to any arbitrary abelian finite group Over multiplicative group in $GF(p)$ and $GF(2^n)$, this method has been successfully applied whereas in case of elliptic curve group it is not yet shown to be applicable This justifies the superiority of elliptic curve cryptosystems over other DLP based cryptosystems To see how this algorithm works, please refer [Men93a, Sim91]

The complexity of this algorithm is given by

$$L[m, a, c] = O(\exp((c + O(1))(\log m)^a (\log \log m)^{1-a}))$$

where c is a constant and $0 \leq a \leq 1$ This algorithm poses a serious attack to cryptosystems which are based on DLP in $GF(p)$ and $GF(2^n)$ To avoid this type of attack, the field size should be of the order of at least 2^{700}

3.2.4 MOV Reduction Attack : An Attack on Elliptic Curve Cryptosystems Only

Since we are concentrating mainly on elliptic curve cryptosystems, we will explain this attack in some detail for ensuring a secure cryptosystem This method which attacks specifically on the security of the elliptic curve cryptosystems only, is given by three cryptographers, Menezes, Okamato and Vanstone [Men93a, Men93b] and hence called as MOV attack The method reduces the DLP in elliptic curve group $GF(q)$ to the DLP in a suitable finite extension $GF(q^k)$ of $GF(q)$ using Weil Pairing method [Men93a, Men93b, Sil85] For this attack to be applicable the multiplicative group of extension field $GF(q^k)$ must be divisible by the order of the elliptic curve defined over $GF(q)$ If $\#E(GF(q) = m$ then following condition must be satisfied

$$q^k \equiv 1 \; mod \; m$$

Now we first define Weil pairing [Sil85, Men93a, Men93b] Let l be a positive integer relatively prime to q Then the Weil pairing e_l is a function

$$e_l \quad E[l] \times E[l] \longrightarrow GF(q^k)$$

where E[l] is subgroup of elliptic curve group of order l Some important properties of the Weil pairing function e_l, are

1 Identity For all $P \in E[l], e_l(P, P) = 1$

2 Alteration For all $P, Q \in E[l]$, $e_l(P, Q) = e_l(Q, P)^{-1}$

3 Bilinearity For all $P, Q, R \in E[l]$

   $e_l(P + Q + R) = e_l(P, R)e_l(Q, R)$ and

   $e_l(P, Q + R) = e_l(P, Q)e_l(P, R)$

4 If $E[l] \subseteq E(GF(q))$, then $e_l(P, Q) \in GF(q)$ for all $P, Q \in E[l]$

Now let P be a point in $E(GF(q))$ of order l such that $\gcd(q, l) = 1$ and $R \in < P >$ Then computation of elliptic curve discrete logarithm s such that $R = sP$, can be computed by using following algorithm

Algorithm

1 Begin

2 Find the smallest extension degree k such that $E[l] \subseteq E(GF(q^k))$

3 Find $Q \in E[l]$ such that $\alpha = e_l(P\ Q)$ and $\alpha^l = 1$

4 Compute $\beta = e_l(P\ Q)$

5 Compute s, the discrete logarithm of β to the base α in $GF(q^k)$

6 End

This algorithm requires the minimum field extension in which $E[l]$ can be embedded If k is large then the algorithm takes long time in finding discrete logarithm as the size of the extension field becomes very large Since there are few isomorphic classes of super singular elliptic curves and for each class the order of the curve is known, k can easily be found It is found that k is 4 and 6 for $GF(2^n)$ and $GF(p)$ respectively Hence this method of computing discrete logarithm in elliptic curve group poses a serious threat of super singular based elliptic curves based cryptosystems Since there are plenty of choices for the order of the non supersingular curve over a given finite field the curves can be so selected that MOV attack becomes infeasible This is the reason why non supersigular curves are being looked with a great interest in the field of cryptography

In the next section we will see various aspects regarding the selection of an elliptic curve to build a secure cryptosystem

3 3 How Elliptic Curves Should be Selected Against all Possible Attacks

In the previous section, we had been familiar with all possible attacks that can break the security of an elliptic curve cryptosystem Here we will see all possible solutions to make these attacks infeasible with the current computational resources and capabilities

To make the square root attack infeasible, the order of the elliptic curve group must be greater than a 30 digits decimal number While to make the Pohlig Hellman attack infeasible, the order of the curve must contain a large prime factor

Hence to avoid both these attacks the order of the curve must contain a prime factor greater than 30 decimal digits number The index calculus attack is impractical over elliptic curves so we should not worried about this attack

Now we discuss the possible solutions to make MOV reduction attack infeasible for the curves over $GF(2^n)$ Let us assume that $\#E(GF(2^n)) = m = c{*}p$ where c is a small number(say less than 100) and p is the large prime factor (greater than 30 decimal digits) Since there are 7 and 3 isomorphism classes of super singular elliptic curves over over $GF(2^n)$ for even and odd n respectively the MOV reduction attack is very much effective for these curves The maximum value of the minimum degree of extension k is only 4 To make this attack infeasible the field $GF(2^n)$ must be so chosen that the order of the curve contains a prime factor greater than 30 decimal digits and extension field $GF((2^n)^k)$ is larger than $GF(2^{700})$

For non supersingular curves, Miyaji s variation of MOV algorithm [Miy91] is applied for finding the discrete logarithm In this modified algorithm, the DLP over non supersingular curve (always having an even order) is mapped to the DLP in the subgroup of the given curve group which has an odd order Remember that $m = c * p$ is even for non supersingular curve Let c' be odd part of c If the prime factor p is such that $(p-1)/2$ is B nonsmooth (i e $(p-1)/2$ has no factor less than B) and

$$(2^n)^{2\phi(c')} \not\equiv 1 \; mod(c'p)$$

then k will be greater than B

Hence we conclude this discussion with a result that to obtain a secure elliptic curve cryptosystem against all possible attacks the order of the selected elliptic curve must contain a prime factor of at least 30 decimal digits and the extension degree for MOV attack must be controlled by a lower bound

3 4 Cryptographic Implications

The discrete logarithm problem in F_p, p a 192 bit prime, has been recently computed by La Macchia and Odlyzko [MaOd91] using index calculus method But it does not seem to be practical for F_p where $p \leq 2^{512}$ Gordon and McCurely [GoMc92] recently indicated that computing discrete logarithms in F_{2^m} for m about 500 is barely feasible given large amount of computer resources Therefore it appears that given the best algorithms known for the discrete logarithm problem in finite fields and given the best available computer technology the discrete logarithm is infeasible for finite fields of size greater than 2^{600}

Now we comment on the security aspects of a family of super-singular curves defined as,

$$y^2 + y = x^3 + b \text{ over } F_{2^m} \quad m \text{ odd}$$

that has previously been suggested for the implementation of elliptic curve cryptosystems Since the k value of MOV reduction attack for these curves is 2, the DLP in these curves is efficiently reducible to the DLP in the quadratic extension of the given field A particular member of this family given as,

$$E \quad y^2 + y = x^3$$

is especially attractive for implementation purpose It is now clear that using E over F_{2^m} is no more secure than using the cyclic group in $F_{2^{2m}}$ Since the cost of computations on the curve is higher than that of in $F_{2^{2m}}$, such a curve is inferior for cryptographic purposes to other existing systems This curve was first considered for the implementation purpose by Koblitz [Kob87] with the particular values $m = 61$ and $m = 127$ These curves are obviously inadequate for cryptographic purposes, since DLP in the fields $F_{2^{122}}$ and $F_{2^{254}}$ is very much feasible Later $m = 191$ and $m = 251$ were suggested but these curves should also be avoided for the same reasons Alternative to the curve $y^2 + y = x^3$ are the super singular curves $y^2 + y = x^3 + x$ and $y^2 + y = x^3 + x + 1$ over F_{2^m} m odd These curves have k value equal to 4

With this discussion, we conclude this chapter In this chapter, we discussed various methods for breaking the discrete logarithm problem Simultaneously, we derived necessary and sufficient conditions for chosen elliptic curve to make all these methods infeasible In the next chapter, we will see that how elliptic curve cryptosystems are built by employing elliptic curve discrete logarithm problem Our focus will be mainly upon the designing of an elliptic curve smart card cryptosystem

Chapter 4

Some Elliptic Curve Public Key Schemes and Design of an Elliptic Curve Based Smart Card Cryptosystem

In the previous chapter, we discussed various security aspects of elliptic curve discrete logarithm problem and all possible solutions for this problem to be unbreakable Here, we will see that how this discrete logarithm problem is employed in designing of cryptosystems Remember that the designed cryptosystems to be secure, the selected elliptic curve must satisfy all the conditions derived in the previous chapter To understand design and setup of these systems, we shall discuss the elliptic curve analogs of some well known public key schemes and then we will design an elliptic curve based smart card cryptosystem

4.1 Elliptic Curve Analog of Diffie-Hellman Key Exchange Scheme

Private key cryptosystems on the one hand work faster but on the other hand they suffer with the key distribution problem as mentioned in the chapter 1. Public key cryptosystems do not have any key distribution problem but they work slower. Keeping this in mind Diffie and Hellman in 1976, gave a public key scheme [DH76] to share a common secret key over an insecure communication channel. Later, this secret key can be used in a private key cryptosystem such as DES. The cryptosystem is hence, called hybrid cryptosystem. Diffie and Hellman gave this key exchange scheme over any arbitrary group. we will describe this scheme in terms of elliptic curve group.

Suppose A and B want to share a common secret key over an insecure channel. For this, they first publicly choose a finite field F_q and an elliptic curve E defined over it. Then they publicly choose a point $P \in E$. To generate a common secret key, A chooses a random integer a which he keeps secret. Now he computes $aP \in E$ and transmits aP to B over a public communication channel. Similarly, B generates a secret random integer b, computes $bP \in E$, and transmits bP to A over the same channel. A receives bP and computes $a(bP) \in E$. B recieves aP and computes $b(aP)$. In this way, both A and B share a common key given by abP. It is clear that without solving the discrete logarithm problem there is no way to compute abP knowing only aP and bP.

4.2 Elliptic Curve Analog of ElGamal Scheme over F_q, $q = p^r$, for Message Encryption and Decryption

An elliptic curve analog of Elgamal scheme was first suggested by Koblitz [Kob87]. Consider an elliptic curve $E(F_q)$ defined over F_q with order $\#E(F_q)$. Let $P \in E$ be a fixed and publicly known point Each user chooses an integer k_i randomly, such that $0 < k_i < \#E(F_q)$ and makes point $k_i P$ public while keeping k_i secret

Message Imbedding

The mapping of plaintext messages to some points on the working curve is the *Message Imbedding* Before encrypting the messages are made into blocks and each block is suitably related to a point on the curve This has to be done in a simple systematic way, so that the plaintext m which is an integer in some range can readily be determined from the knowledge of the coordinates of the corresponding point P_m Koblitz gave a probabilistic method to imbed plaintexts as points on an elliptic curve E defined over F_q, where $q = p^r$ is assumed to be large For knowing about this method, refer [Kob87 Kumar96]

Encryption/Decryption

Let A and B are two users having the private keys k_1 and k_2 respectively The public key of A is k_1P and of B is k_2P If A has to send a mapped message P_m to B he computes and transmits the following ciphertext to B

$$C = (k_1P, \ P_m + k_1(k_2P))$$

So the computations involved in encryption are k_1P and $k_1(k_2P)$ Remember that k_1, the encryption key, is not fixed It is chosen at the time of encryption

At the recieving end B multiplies the first part of the recieved ciphertext, i e k_1P by his secret key k_2 and obtains $k_2(k_1P)$ This computed term now he subtracts from the second part of the recieved ciphertext and gets the original message P_m as shown below

$$P_m = P_m + k_1(k_2P) - k_2(k_1P)$$

In this way, we see that the security of this system relies on the elliptic curve discrete logarithm problem

4 3 Elliptic Curve ElGamal Cryptosystems over F_{2^m}

These are the systems which we have implemented in this thesis Implementation of ElGamal scheme over F_{2^m} [MV90, MV93] has a slightly different approach Now we will see how this approach works Let us consider a non supersingular curve $E \quad y^2 + xy = x^3 + a_2x^2 + a_6$ defined over F_{2^m} and let P be a publicly known point on E Assume that the elements of F_{2^m} are represented in **normal basis** [LdNd] The advantages of this assumption we will explain in the chapter based on efficient implementation issues Well, user A randomly chooses an integer a and makes public the point aP, and keeps a secret To transmit the message pair $(M_1 \ M_2)$ to A sender B selects a random integer k and computes the point kP and $k(aP) = (\bar{x}, \bar{y})$ Since the event $\bar{x} = 0$ or $\bar{y} = 0$ occurs with negligible probability for randomly choosen k we can assume $\bar{x} \ \bar{y} \neq 0$ B then sends A the point kP and the field elements $M_1\bar{x}$ and $M_2\bar{y}$ At the recieving end to read the message, reciever A multiplies the first part of the ciphertext kP by his secret key a and obtains $a(kP) = (\bar{x}, \bar{y})$ from which he can recover M_1 and M_2 by dividing the recived $M_1\bar{x}$ and $M_2\bar{y}$ in two divisions

Based on this scheme, we have made a processor and operating system independent software package for encryption/decryption over $GF(2^{113})$ We have also made an effort to implement this scheme over much higher field, upto $GF(2^{400})$, on the Texas' digital signal processor

4 4 Elliptic Curve Analog of RSA

In broadcast applications the conventional RSA system, based on the integer factorization problem, is not secure if the encryption key e is small this was first shown by Hastad, in 1985 Actully he presented a paper [Hast85] in which he shown that an attack based on the Hastad theorem, called the *low exponent attack* is very effective against the conventional RSA Later it has been shown in [KK94,KOT94] that RSA type cryptosystems over elliptic curves such as the KMOV and Demytko cryptosystems are more secure than the original RSA against the low exponent attack or more clearly *low multiplier attack* in terms of elliptic curve terminology. To know more about the working of these systems and about low exponent attack the interested readers may refer to [Kumar96].

4.5 An Elliptic Curve Based Smart Card Cryptosystem

The more our society becomes computerized, the greater are the risks from banking fraud, economic sabotage, industrial spying etc. An obvious conclusion is that our computerized open systems require additional security. Cryptography is a powerful security tool in the field of information technology. However, the expansion of public cryptologic knowledge is moderated by goverment and political concerns aiming at controlling the spread of cryptologic technology and devices, expressed most often in the form of embargos or export controls.

The smart card, which stores, processes, and controls internal cryptographic algorithms, [GuUg86, HaWa88] as we will see, suggests solutions that may satisfy both national regulations and commercial needs. Smart cards are already in widespread public use. Through this user friendly technology, cryptology is invading our everyday life. This invasion has a large influence on security in various fields of applications, not only in banking, but also in the areas of health, pay television, telephone, home computers, data processing, communication network, and more generally, information technology. This pocket size computer is so popular today that some consider it to be a fourth level in the hierarchy after the host computer, the departmental computer, and the personal computer.

4 5 1 What a Smart Card is

At first glance a smart card appears to be simply an improved traditional credit card But a smart card is in reality a multipurpose tamper resistant security device which is capable of doing cryptographic operations like encryption, decryption and digital signatures All the computations are done on the card itself without requiring any external computing resources This guarantees that all the secret informations are kept on the card itself, they never leave the card Because of all these features the smart cards are highly reliable devices as far as security is concerned

Traditional financial cards or more simply ATM cards are magnetic strips cards which do not have any processing capability They can only store the data Smart cards on the other hand can process the data as well in addition to store it Actually, this pocket size plastic card possess a single 20 mm^2 VLSI chip which is reponsible for doing various cryptographic operation This idea of inserting a chip into a plastic card is not new but practical public key applications emerged only a few years ago because of previous limitations in the storage and processing capacities of circuit technology

The chip embedded in a smart card is a single chip microcomputer (MCU) A MCU is a computer system integrated onto a single piece of silicon The only computerlike resources it lacks are the external human interface devices such as keyboards displays, disk drives, etc The major difference between this MCU and a general purpose MCU is that a general purpose MCU can be used in any operating mode selected by the user and internal data and address buses can be accessed from outside hence internal contents can be changed While, with MCU, designed for smart card, the only possible operating mode is the use mode After the device has been tested and passed as fully functional by the manufacturer, the users can only use it under the exclusive control of the user software in the on board ROM The internal buses are never accessible from outside

4 5 2 The Internal Architecture of a Smart Card

To illustrate that what are the various components a smart card must have to perform the required task we will discuss in this section smart card IC chip in some detail

A smart card IC is consructed from predefined logic modules as shown in thc figure ' Smart Card Microarchitecture ' Before passing it finally to the market each and every module must be tested for its inputs, outputs, operations and security capability

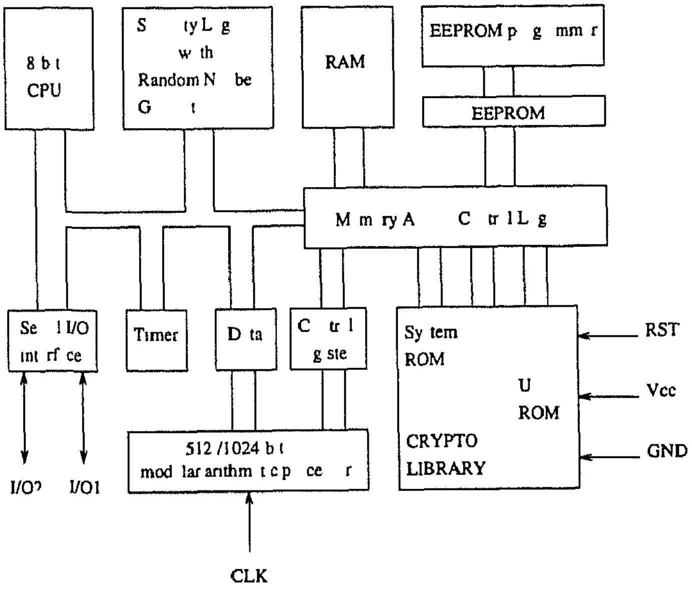

Figure 4 1 Smart Card Microarchitecture

The major part of a smart card IC is its on chip memory The memory can be divided into four distinct areas RAM, system ROM user ROM, and user EEPROM The amount of RAM and ROM can vary from card to card depending upon the type of application that the card has to handle The typical figures are given in the table

RAM	128 960 bytes
System ROM	1 8 KB
User ROM	6 24 KB
User EEPROM	256 bytes 8 KB

Each memory area has a user defined memory access contol logic, which provides full separation between on chip application code and the data

The system ROM holds basic I/O, test and security function These functions are based on a firmware library that maximizes security The user ROM holds the operating system code This code will differ depending upon the requirements of various end applications Since both the system and the user areas are ROM their code contents are placed in the device at the time of manufacturing so that they are fixed for the lifetime of the smart card The user EEPROM area stores variable data such as personal data keys, a purchase history, perhaps your Social Security number and favorite telephone numbers It is partitioned into several zones corresponding to different uses and accesses modes from the outside of the card

- The SECRET ZONE is impossible to read from outside the chip, either by logical or physical way

- The ACCESS ZONE is a service one in which are memorized access using the issuer s or the user's keys This memory allows to count good or wrong submitted keys, and to lock the circuit in case of several wrong submitted keys (usually three)

- The CONFIDENTIAL ZONE can be read if the good issuer s or user's key is given It contains generally personal or sensitive data

- The TRANSACTION ZONE is used during the current life of the card This can be read or written, with or without a key according the application

The CPU designed for smart card is usally 8-bit microcontroller but 32-bit is also under developement The CPU features extended addressing modes and an instruction set that is especially designed for writing high security applications The card's microcontroller executes the cryptographic application programs which are stored in ROM at the time of the manufacturing. The most common cores being Motorola's 68HC05 and Intel's 80C51. Recently, SGS Thomson also developed a low power logic smart card IC ST19.

For cryptographic applications, the modular arithmetic processor provides public key cryptography calculations using up to 512 bit or 1024-bit keys. When this is used together with the on chip random number generator, the IC can perform full public key generation, digital signatures, and authentication internally. This capability guarantees that the secret key will never be known outside the smartcard and contributes to the overall security of the system.

4.5.3 About Smart Card Software

No CPU based system is ever complete without a firmware developement system. This, too, is provided both as a complete developement system and as Crypto Library support routines. The Crypto Library can be provided in the system ROM area, leaving 6 to 24 KB of user ROM available for the application software.

The cryptographic library provides firmware functions for:

- Basic math, including modular squaring and multiplication for various length digits
- Generating long random numbers
- Calculating Montgomery constants, which are required for long-number modular arithmetic
- Modular exponentiation
- More elaborate functions such as RSA signatures and authentication for any modulo length upto 1024 0r 2048 bits (depending on the required security level)
- Full internal key generation for signatures and authentications

4 5 4 High-level implementation of Smart card

How an user can get various cryptographic services from the card we will see this now During high level implementation it is generally recommended to separate the cryptographic scheme (RSA DSA and so on) from the cryptographic operations (sign verify encrypt decrypt hash and key exchange) Designers can achieve this by implementing an I/O buffer in the card into which the terminal writes data to be processed In this model the following steps are executed whenever the card performs a cryptographic operation

1 A put command selects a key file specific to a scheme

2 A put command writes data to process (message ciphertext signature, and so on) to the I/O RAM buffer

3 A get command (specific to an operation) retrieves the card s result

Such an approach results in a simplified command set and allows upgrading of the card without adding new command codes The following toy example illustrates

- the encryption of the message "process me that' with the RSA keys contained in the file 2401,
- the signature of the message "123" by DSA file 334A, and
- a Diffie Hellman key exchange with the keys contained in the file E1F3

select file 2401	exchange card to reader
{RSA, 768, s/e/i}	/*≅ TYPE in DOS */

File selection returns the key file type (here 768 bit RSA) and the cryptographic operations allowed with this file (s=signature and verification, e=encryption and decryption, i=identification, k=key echange)

put data	exchange reader to card

```
{ process me that'}          /* data to process */

get data
encrypt 0000                   exchange  card to reader
{"E32A371B908AB37"}           /*≅ENCRYPT EXE */
```

The 0000 sent to the card means that the result (here the ciphertext beginning E32) should be sent to the terminal A nonzero code would indicate a file ID where the ciphertext should be written

```
select file 334A        exchange  card to reader
{DSA  512, s}           /*≅ TYPE in DOS */
put data                exchange  reader to card
{ 123' }                /* data to process */
get data
sign 0000               exchange  card to reader
{"ADE603B826FDE04"}     /*≅ SIGN EXE */
select file E1F3        exchange  card to reader
{D H, 512, k}           /*≅ TYPE in DOS */
put data                exchange  reader to card
{ process me that"}     /* a^x mod p */
get data
key exchange 2010       exchange  card to reader
{'AE589EB6A564CDD"}     /*≅ KEY EXCH EXE returns a^y mod p */
```

During a key exchange, the user must specify a destination file ID (here 2010) for the common key The outside world can never access this value

4 5 5 What can a Smart Card Do in a Cryptosystem

In a cryptosystem, the main purpose of the smart card is to authenticate the card holder to a system which is located at a remote place [sch89, Miy92] The smart cards are first initialized with proper keys so that a secure communication can be done using a cryptographic algorithm For interfacing with the external world, smart cards are inserted in a device called smart card reader attached to your PC or work station Once a smart card is inserted the following operations are performed

- **User to Smart Card Authetication** The card must be sure that the right card holder is present during some operations For this, every user is given a number called personal identification number (**PIN**) Using this PIN smart card identifies the user The basic drawback with PIN identification is that if it leaks out then the correct identification will not be guaranteed An alternative is, to use biometric techniques, i e voice finger print image etc But these techniques require more memory

- **Smart Card to Remote System Authentication** After the user authentication, the validity of the card to the remote system at the other end has to be checked Whether the inserted card is authorized one or not, this has to be checked by the remotely located system otherwise manufacturing of the fraud card comes into picture

- **Remote System to Smart Card Authentication** Now there is a need for the remote system also to be authenticated to the smart card at the tramsmitting end A similar protocol is run from the remote system to the card

Once, all of these authentication are done successfully, the user is granted access to the system and further communication can take place

Now at this point we have been familiar about a cryptographic smart card Presently almost all the card manufacturer like Motorola, SGS Thomson Hitachi, Gemplus etc are using RSA and ElGamal schemes for encryption and decryption while Schnorr [sch89] for identification and signature The disadvantages of these schemes for smart card viewpoint we have already discussed in chapter 1 Recently Siemens has taken the step for elliptic curve based smart card manufacturing with keeping in the mind the various advantages that elliptic curves offer In the next section we will see that how elliptic curves may be employed for designing smart card cryptosystems

4 5 6 Elliptic Curve Based Smart Card

Here we discuss that an elliptic curve based discrete logarithm problem can be employed in a smart card to perform the tasks mentioned in the previous section We will consider the ElGamal scheme over $GF(2^n)$ for various authentication processes involved in the card outside world transaction The message encryption and decryption method have already been given in the Section 4 3 Here we will see the authentication only The process is similar to Diffie Hellman's key exchange protocol

Let P be a publicly choosen point on a publicly choosen elliptic curve E Let u & $U(= uP)$ and s & $S(= sP)$ be the private and public keys of the user and remote system repectively The remote system maintains a data base for the public keys of the users The following information is required to be stored in the card memory

1 Personal Identification Number (similar to password) of the user

2 Card's Identity number

3 The modulo polynomial for $GF(2^n)$

4 The coefficients of elliptic curve equation

5 The coordinates of the point P

6 The size of the order of the curve

7 The secret key u

8 The public keys U and S

User to Card Authentication User inserts the card in a smart card reader and enters his PIN If the PIN matches with that stored in smart card memory, the process proceeds further otherwise terminates

Card to Remote System Authentication The card sends its identity number (not PIN) to the remote system so that the system picks the corresponding public key from its database Now the process proceeds as follows

1 The remote system finds a random integer k and computes kP The kP is sent to the card

2 The card computes $u(kP)$ using user's secret key and sends it back to the remote system

3 The remote system computes kU using user's public key and compares with the received $u(kP)$ If they match then the remote system can be sure of validity of the card

Remote System to Card Authentication A similar protocol runs as follows

1 The card finds a random integer k and computes the kP The kP is sent to the remote system

2 The remote system computes $s(kP)$ using the secret key and sends it back to card

3 The card computes kS using remote system's public key and compares with $s(kP)$ recieved from the remote system If they match then the card can be sure of validity of the system

The microcontoller of the smart card is programmed to perform all the computations required in the above protocols The application program is stored in ROM The EEPROM contains all the information specific to user and algorithm i e public key, private key, algorithm parameter Since the size of the memory is limited, the storage requirements of any public key algorithm is one of the major criteria for its selection for the smart card To minimize the storage requirement, how the field and curve should be chosen for a smart card this we will see in the next section

4 5 7 Selection of Field and Curve for Smart Cards

Since the storage and computation requirements for a smart card based application are the crucial factors, our major concern here will be to minimize the storage for a smart card and also to select a field and curve so that computations involved are reduced

We first concentrate on storage requirements [Miy92] which is very important in hardware implementation Let the selected field be $GF(2^n)$ The information to be stored and corresponding storage requirement in bits is given below

- The irreducible polynomial over which the field is selected The corresponding storage requirement is $n + 1$ bits
- Curve coefficients a and b for the curve $y^2 + xy = x^3 + ax^2 + b$ The corresponding bits to be stored are $2n$ bits
- The base point P (x, y) The corresponding storage requirement is $2n$ bits
- Secret key For this at the most n bits are needed
- Public key of self and the remote system as well The total $4n$ bits are needed for this
- Size(number of bits) in the order of the curve
- PIN and card identification number

As we will see in the next chapter that **optimal normal basis** are very attractive for hardware implementation of $GF(2^n)$ as the squaring and addition each can be performed in one clock cycle and multiplier has the minimal complexity Unfortunately, the ONB does not exist for every extension of $GF(2^n)$ But if the ONB exists in $GF(2^n)$ then it is uniqe and corresponding multiplier will also be uniqe irrespective of the modulo polynomial used for defining the field $GF(2^n)$ Once we know the ONB representation of any element of $GF(2^n)$ the modulo polynomial will not be required to be stored Hence there is a memory saving of $n + 1$ bits

If a non supersingular elliptic curve is selected then addition formulae require only coefficient a Hence only coefficient a needs to be stored The addition of two points on supersingular curves requires a and b The size of the secret key will approximately be the same as that of field The base point P requires storage of two $GF(2^n)$ elements The two public keys require storage of 4 elements of $GF(2^n)$ It may not be always necessary to store the public keys in the smart card if all the keys can be stored in a common database which contains all the public keys with certification [Sta95] The storage requirement for the size of the order is insignificant Similarly PIN and card s identity number are always required Hence we see that an elliptic curve based smart card requires $8n$ bits or $9n$ bits of storage as the curve is non supersingular or supersingular

4 5 8 Comparison with RSA based Smart Card

As we discussed in the previous section the elliptic curve based smart card needs $8n$ or $9n$ bits of storage if non supersingular or supersingular elliptic curves are used If the selected field is $GF(2^{130})$ then the total storage requirements is roughly 1K bits or 1 1K bits Now we make an estimate of the storage requirement of RSA based smart cards for the sake of comparison This comparison is necessary because RSA is the most widely used algorithm for smart card in the present scenario and in future, elliptic curve public key algorithm may become an alternative for it

As we discussed earlier, RSA cryptosystem requires at least 512 bit modulus For the 512 bit modulo RSA based smart card, the information to be stored and corresponding storage requirement in bits is given below

- Publicly known modulo integer n_u obtained by the card This requires 512 bits to be stored

- Publicly known modulo integer, n_s, obtained by the remote system This requires again a 512 bit storge

- Public key of the user which is a 512 bit long enciphering exponent e_u

- Public key of the remote system which is again a 512 bit long enciphering exponent e_s
- Secret key of the user which is a 512 bit long deciphering exponent d_u

In this way the storage requirement is approximately 2560 bits which is more than double of what is required for an elliptic curve based smart card Hence we can see that elliptic curve based smart card requires less memory in comparison to RSA based smart card

From the view point of required computations RSA public key algorithm needs larger number of computations as compared to elliptic curve algorithm Hence the complexity of the coprocessor of an RSA based smart card is far more than that required for an elliptic curve based smart card The major computation in an elliptic curve public key algorithm is the computation of multiple of a point If the private key k contains x bits with the Hamming weight y then the computation of kP requires x doublings and y additions For a non supersingular elliptic curve over $GF(2^n)$ addition of two distinct points takes two field multiplications and one inversion, while doubling a point takes three multiplications and one inversion Furthermore, one inversion takes $\lfloor \log_2(n-1) \rfloor + \omega(n-1) - 1$ field multiplications where $\omega(n-1)$ is the number of ones in the binary representation of $n-1$ If the chosen field be $GF(2^{130})$, inversion takes 8 multiplications and hence addition and doubling take 10 and 11 multiplications respectively This shows that the total number of field multiplications to compute kP is $11x+10y$ For $GF(2^{130})$, the typical values of x and y will be 130 and 130/3 With these values of x and y, number of field multiplication approximately equals 1864 Since, for $GF(2^n)$, each multiplication in ONB form needs n clock cycle (for hardware multiplier), total number of clock cycles to compute kP over $GF(2^{130})$ approximately equals 242320 Moreover, number of field multiplications can be reduced by using projective coordinates instead of affine coordinates In this case, computation of kP requires $7x + 13y$ multiplications because addition and doubling of points requires 13 and 7 multiplications repectively This shows that the expected number of clock cycles in computation of kP will be

$7nx + 13ny$ With the above values of x and y for $GF(2^{130})$ the number of field multiplication equals 1482 and the number of clock cycles required in computation of kP will approximately be 192660 In an equivalent RSA cryptosystem with 512 bit long modulo integer to encipher or decipher a 512 bit number requires 1 2 multiplications in modular arithmetic per bit or about 750 multiplications total A hardware chip [WQ90], designed for RSA based smart card has already come in the market that could compute m^e mod n, with 512 bit operands, in less than 1 5 seconds (6 MHz) requiring approximately 9000000 clock cycles These estimates clearly depict the superiority of elliptic curve cryptosystems

Hence we see that elliptic curves are a better choice for smart card as compared to presently used RSA Since RSA with 512 bit modulus will face a serious security threat in near future, these cryptosystems with larger modulo integers are also being realized which will further increase the memory and computation requirements In such a scenario, the elliptic curve based smart card will definitely be a better alternative for RSA based smart card

With this we conclude this chapter For more information on smart cards please see [Sim91 NM95 Kon91, FOM92, DVJ96 AMV93] In the next chapter we will concentrate on the actual software implementation

Chapter 5

Implementation and Results

5.1 Introduction

Up to now, we discussed almost all the fundamentals of the elliptic curves which are required for building a secure elliptic curve public key cryptosystem. Since an elliptic curve cryptosystem offers various advantages over other existing public key cryptosystems, the practical implementation of these systems is becoming a subject in which most of the cryptographers are taking interest today. Their continuous efforts have already resulted in some useful hardware implementation in the form of application specific integrated circuits [AMV93]. Although hardware implementations provide much higher throughput, but in most of the network and internetwork applications, software implementations are generally preferred over hardware implementations. Keeping in the mind the flexibility and other advantages of software implementation, a lot of work is being done in this area as a challenge for cryptographers and software programmers [HMV92, Kumar96, Pank97]. In this thesis, we have independently tried to implement elliptic curve public key cryptosystems in software. Here we are not claiming this to be efficient since any implementation is machine and code dependent. But the main advantage of this implementation is that all the programming has been done using today's most general purpose high level language ANSI C without using any readymade package available for cryptography (such as SIMATH). This approach is very useful at application level where installing the whole package is not the right suggestion for getting small cryptographic applications like e mail security, FAX security and smart card microcontroller programming Recently, two largest manufacturer of smart card, Gemplus and Schlumberger announced the idea of JAVA programming for smart card For more information refer [Gemp]

The implementation done in this thesis can be divided in two parts In the first part, an operating system and processor independent package for message encryption and decryption, based on elliptic curve ElGamal scheme over $GF(2^n)$, has been written This may be useful for Email and Fax security applications The field used is $GF(2^{113})$ In the second part, implementation of encryption decryption and authentication algorithms over TMS320C40 25 MHz digital signal processor has been done The approach used here is based on optimizing C cross compiler which takes the C programs as input and generates the optimized assembly language programs as output The size of the field used here can be upto $GF(2^{400})$

5 2 Why We Have Used Normal Basis Representation in Our Implementation

For any chosen field $GF(2^n)$, there are many different bases to represent its elements Two of them are standard or polynomial basis and normal basis With keeping in the mind the advantages of normal basis representation, we have used this representation for all the field elements in our implementation A normal basis of $GF(2^n)$ over $GF(2)$ is a basis of the form

$$\{\beta, \beta^2\ \beta^{2^2}, \quad , \beta^{2^{m-1}}\}$$

where $\beta \in GF(2^n)$, it is well known [LdNd] that such a basis always exists Given any element $\alpha \in GF(2^n)$, we can write

$$\alpha = \sum_{i=0}^{n-1} a_i \beta^{2} \text{ , where } a_i \in \{0, 1\}$$

Since squaring is a linear operator in $GF(2^n)$, we have

$$\alpha^2 = \sum_{i=0}^{n-1} a_i\beta^{2^{i+1}} = \sum_{i=0}^{m-1} a_{i-1}\beta^{2^i} = (a_{m-1}, a_0, \ldots, a_{m-2})$$

with indices reduced modulo n. Hence a normal basis representation of $GF(2^n)$ is advantageous because squaring a field element can then be accomplished by a simple rotation of the original representation, an implementation that is easily implemented in hardware as well as in software.

Multiplication in a normal basis representation is more complicated. Let $A = (a_0, a_1, \ldots, a_{n-1}), B = (b_0, b_1, \ldots, b_{n-1})$ be arbitrary elements in $GF(2^n)$, and let $C = A \cdot B = (c_0, c_1, \ldots, c_{n-1})$. Then

$$C = \sum_{k=0}^{n-1} c_k\beta^{2^k} = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_j\beta^{2^i}\beta^{2^j} \qquad (a)$$

If we let

$$\beta^{2^i}\beta^{2^j} = \sum_{k=0}^{n-1}\lambda_{ij}^{(k)}\beta^{2^k}, \quad \lambda_{ij}^{(k)} \in \{0, 1\}, \qquad (b)$$

then comparing coefficients of β^{2^k} in (a) yields the formulae

$$c_k = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_j\lambda_{ij}^{(k)}, \quad 0 \le k \le n-1 \qquad (c)$$

Raising both sides of (b) to the 2^{-l} th power, we find that

$$\beta^{2^{i-l}}\beta^{2^{j-l}} = \sum_{k=0}^{n-1}\lambda_{i-l,j-l}^{(k)}\beta^{2^k} = \sum_{k=0}^{n-1}\lambda_{ij}^{(k)}\beta^{2^{k-l}} \qquad (d)$$

Equating coefficients of β^{2^0} in (d) then yields

$$\lambda_{ij}^{(l)} = \lambda_{i-l,j-l}^{(0)}, \quad for\ \ all\ \ 0 \le i, j, l \le n-1$$

The formula (c) can now be rewritten as

$$c_k = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j \lambda_{i-k\,j-k}^{(0)} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{i+k} b_{j+k} \lambda_{ij}^{(0)}$$

Hence if a logic citcuit with input A and B is built to compute the product digit c_0 the same circuit with inputs $A^{2^{-k}}$ and $B^{2^{-k}}$ yields the product digit c_k Note that $A^{2^{-k}}$ and $B^{2^{-k}}$ are simply cyclic shifts of the A and B In this way the use of normal basis simplifies the computations involved in the implementation of the multiplier

The *complexity* of such implementation is determined by C_N the number of non zero terms $\lambda_{ij}^{(0)}$ Clearly we have $C_N \leq n^2$ A lower bound on C_N is $C_N \geq 2n-1$ [MOVW88] If $C_N = 2n-1$, then the normal basis is said to be *optimal normal basis* ONB So, if we use optimal normal basis for field elements representation then the multiplier attains the minimal complexity Unfortunately the ONB does not exist for every extension of $GF(2)$ The following two theorems tell us about the existence of ONB in $GF(2^n)$

Theorem 1 If $n+1$ is prime and 2 is primitive element of $GF(n+1)$, the $(n+1)$ th roots of unity in $GF(2^n)$ form the ONB

Theorem 2 If $2n+1$ is prime and either

1 2 is primitive in $GF(2n+1)$, or

2 $2n+1 \equiv 3 \ mod \ 4$ and 2 generates the quadratic residues in $GF(2n+1)$

then for $(2n+1)$ th root of unity β in $GF(2^{2n})$ $\gamma = \beta + \beta^{-1}$ will be the optimal normal basis generator

So, in order to attain the minimal complexity in the implementation, the selected field $GF(2^n)$ should be such that n satisfy any of the conditions given in these two theorems In our case, we have taken n according to the theorem 2 Some useful values of n for which ONB can be constructed in $GF(2^n)$ are 100, 105, 106, 113, 119 130, 131, 134 135, 138, 146, 148, 155, 158, 162, 172 173, 174, 178, 179, 180 183

186 189, 191 194, 196, 209 210 221 and so on Since these values are sufficient for elliptic curve cryptosystems to be secure, that s why we are not giving higher values of n For More values of n, readers may refer to [MOVW88]

In the following section we will discuss various steps in sequel which have been involved to reach on the final implementation At each and every step the corresponding procedure for efficient implementation is also given

5 3 Various steps involved in the implementation

It is clear now that to implement an elliptic curve public key cryptosystems, the basic computation involved is of kP where k denotes the private key and P is the publicly known base point on the chosen elliptic curve Since computation of kP means k times addition of the point P, the only operations involved to compute kP are addition of two points and double of a point Explicit formulae for addition of two points and double of a point have already been given in chapter 2 It is clearly visible from these explicit formulae that the arithmetic operations involved in these formulae are squaring, addition, multiplication and inverse Since our implementation is based on optimal normal basis squaring is just a simple rotation, addition is just a simple exclusive OR There is an efficient algorithm available to compute the inverse in terms of multiplications, as we will see later on So the only basic operation needed to compute kP is the multiplication of two field elements

As we saw in the previous section, to implement a normal basis multiplier it is necessary to compute the lambda matrix denoted as $\Lambda = (\lambda_{ij})_{n \times n}$ This matrix is fixed for a given field $GF(2^n)$ and the number of ones in this matrix equals $2n - 1$ We could successfully implement this matrix for the given field $GF(2^n)$ How this could be achieved is shown below in the various steps

5 3 1 Computation of Lambda Matrix Required for Normal Basis Multiplication

To get $\Lambda = (\lambda_{ij})_{n \times n}$ we adopt the following efficient procedure

1 Find the minimal polynomial of ONBG denoted as $m_\beta(x)$ This is the irreducible polynomial over which the chosen field $GF(2^n)$ is defined

2 Find all the products $\beta^{2^0}\beta^{2^j}$ $0 \leq j \leq n-1$ (n products) modulo $m_\beta(x)$

3 Express the polynomial basis representation of the above products in normal basis representation using the transformation matrix T

4 Order all the above normal basis representations of $\beta^{2^0}\beta^{2^j}$ $0 \leq j \leq n-1$ with increasing j to give the matrix $Q = (q_{ij})$

5 Use the relation $q_{ij} = \lambda_{-j\,i-j}$ to get $(\lambda_{ij} = \Lambda$ from $Q = (q_{ij})$

So to implement this approach to get lambda matrix we have to implement above steps as given in the following sections

5 3 2 Finding The Minimal Polynomial of ONBG

This is the first step towards lambda matrix Assuming that β is the ONBG according to theorem 2 then it is easy to derive the minimal polynomial of $\beta \rightarrow m_\beta(x)$ and is specified in terms of recursion [Men93a]

Let $f_0(x) = 1$,
and $f_1(x) = x + 1$
then for any $t \geq 2$
$f_t(x) = xf_{t-1}(x) + f_{t-2}(x)$

be the sequence of polynomials $f_i(x)$ $i = 1, 2$ over $GF(2)$ Then if n is such that we have a ONBG gaurenteed, then $f_n(x)$ is the minimal polynomial $m_\beta(x)$ of the ONBG We can clearly recognize that the sequence of polynomials $f_i(x)$ are nothing but Fibbonaci polynomials [McEl] This is a beautiful and quite useful coincidence in that, they are very easy to generate

5 3 3 Finding the multiplication of Two $GF(2^n)$ Elements Represented in Standard Basis

This is the second step towards lambda matrix which we have implemented Let $A = (a_0\ a_1, \quad , a_{n-1})$, $B = (b_0, b_1, \quad , c_{n-1})$ and $C = (c_0, c_1 \quad c_{n-1})$ be their product We have used the following algorithm for computation of C Here $f = (f_0\ f_1 \quad , f_n)$ denotes the coefficients of modulo polynomial $m_\beta(x)$

Procedure GF2PROD(A,B)

```
1 Begin
2 Set $C = (0, 0, \quad , 0)$
3 for $i = 0$ to $n - 1$
      • begin
      • if $(b_{n-1-i} = 1)$ then
        if $(c_n = 1)$ then $C = C + f + A$
        else $C = C + A$
        else if $(c_n = 1)$ $C = C + f$
      • Give a right shift to bits of $C$
      • end
4 End
```

Using this procedure, we could implement to find the first n cross products $\beta^{2^0}\beta^{2^j}$ $0 \leq j \leq n - 1$ modulo $m_\beta(x)$

5 3 4 Finding the Transformation Matrix 'T'

The computation of the transformation matrix is necessary to convert the SB representation into NB representation as

$$NB = T\,SB$$

$\Rightarrow SB = T^{-1}\, NB$

After computation of T, we converted the n cross products as find above into their corresponding normal basis representation according to the third step towards lambda matrix. We have the following procedure to build the matrix T

NB generated by the ONBG β

$\{\beta, \beta^2, \ldots, \beta^{2^{n-1}}\}$

SB generated by the ONBG β

$\{1, \beta, \ldots, \beta^{n-1}\}$

β in terms of $SB = (0, 1, 0, \ldots, 0)$

β in terms of $NB = (1, 0, 0, \ldots, 0)$

we have thus

$$\beta^2 = (0, 0, 1, 0, \ldots, 0)$$
$$\beta^3 = (0, 0, 0, 1, \ldots, 0)$$
$$\beta^{n-1} = (0, 0, 0, \ldots, 0, 1)$$

in terms of SB and hence, if we view any element in $GF(2^n)$ in terms of SB, we can have SB representation of the product, sum of any two elements of $GF(2^n)$ (reduced modulo $m_\beta(x) \rightarrow$ minimal polynomial of β as well as primitive chosen irreducible polynomial). Therefore, we can find the transformation matrix easily by multiplication modulo $m_\beta(x)$.

Using the minimal polynomial of ONBG and procedure GF2PROD(A,B) implemented in step 1 and 2 respectively, we have derived an algorithm for finding the tranpose of the transformation matrix for a given field $GF(2^n)$.

```
Procedure Tmatrix( i)

1 Begin

2 Initialize all $n^2$ elements of $T'$ to zero

3 for($i = 0,\ i < \lceil \log_2 n \rceil,\ i++$)
    T'[i][$2^i$] = 1,

4 for (i = $\lceil \log_2 n \rceil$, i ≤ n 1, i++)
    T'[i] = GF2PROD(T'[i 1], T'[i 1]),

5 end
```

The computed matrix T' from this procedure is the transpose of the desired transformation matrix So we can easily compute the desired transformation matrix T

5 3 5 Standard Basis to Normal Basis conversion

After having the transformation matrix T, the next step we have implemented is standard basis to normal basis conversion This is the simple matrix multiplication program We can find the NB representation as follows

$$[NB]_{n\times 1} = [T^{-1}]_{n\times n}\,[SB]_{n\times 1}$$

where T^{-1} is the invere of matrix T A very efficient and fast matrix inversion program has been written by us

After having this implementation, the step 4 and step 5 are simple mathematical manipulation and we can easily get lambda matrix for the field $GF(2^n)$

5 3 6 Software Implementation of Normal Basis Multiplier over $GF(2^n)$

We have successfully implemented a software version of very efficient normal basis multiplier Since we have chosen optimal normal basis throughout our implementation the designed multiplier is having minimal complexity

In section 5 2 we have already dicussed the multiplication operation of two field elements represented in optimal normal basis Let $\bar{A} = (a_0, a_1 \quad a_{n-1})$ and $\bar{B} = (b_0, b_1, \quad , b_{n-1})$ are two field elements given in ONB representation Let $C = (c_0, c_1, \quad , c_{n-1})$ is their product The procedure for normal basis multilication is as follows Here Λ is the lambda matrix obtained in the previous section

Procedure NBPROD(A,B)

```
1  Begin
2  for ($i = 0, i < n, i++$)
       • Compute $c_i = \bar{A}\Lambda\bar{B}^T$
       • Rotate $A$ by one bit in the left direction
       • Rotate $\bar{B}$ by one bit in the left direction
       • end
3  End
```

In the next section we will see how the inverse operation in the field $GF(2^n)$ can be converted into multiplication operations

5 3 7 Computing the Inverse of an Field Element over $GF(2^n)$

We have implemented the most efficient technique, from the point of view of minimizing the number of multiplications, to compute an inverse of an element in $GF(2^n)$ proposed by Itoh, Teechai and Tsujii [ITT86] Observe that if $\alpha \in GF(2^n)$, $\alpha \neq 0$, then

$$\alpha^{-1} = \alpha^{2^m-2} = \left(\alpha^{2^{n-1}-1}\right)^2 \qquad [\text{since } \alpha^{2^m-1} = 1]$$

If m is odd, then since

$$2^{m-1}-1 = \left(2^{(m-1)/2}-1\right)\left(2^{(m-1)/2}+1\right),$$

we have

$$\alpha^{2^{m-1}-1} = \left(\alpha^{2^{(m-1)/2}-1}\right)^{2^{(m-1)/2}+1}$$

Hence it takes only one multiplication to evaluate $\alpha^{2^{m-1}-1}$ once the quantity $\alpha^{2^{(m-1)/2}-1}$ has been computed (we are ignoring the cost of squaring). If m is even, then we have

$$\alpha^{2^{m-1}-1} = \alpha^{2(2^{(m-2)/2}-1)(2^{(m-2)/2}+1)+1},$$

and consequently it takes two multiplications to evaluate $\alpha^{2^{m-1}-1}$ once $\alpha^{2^{(m-2)/2}-1}$ has been computed. The procedure is then repeated recursively.

This algorithm to compute inverse can be understood more easily by the following example. Consider the field $GF(2^{155})$. We have

$$2^{155}-2 = 2\,(2^{77}-1)\,(2^{77}+1),$$
$$2^{77}-1 = 2\,(2^{19}-1)\,(2^{19}+1)\,(2^{38}+1)+1$$
$$2^{19}-1 = 2\,(2^{9}-1)\,(2^{9}+1)+1,$$
$$2^{9}-1 = 2\,(2-1)\,(2+1)\,(2^{2}+1)\,(2^{4}+1)+1,$$

and so an inversion in $GF(2^{155})$ takes 10 multiplications.

It can easily be verified by induction that this method requires exactly $I(m) = \lfloor \log_2(m-1) \rfloor + \omega(m-1) - 1$ field multiplications, where $\omega(m-1)$ denotes the number of 1's in the binary representation of $m-1$.

5.3.8 Computing the Sum of Two Points

After implementing the multiplication and inverse operation, the next step we implemented is addition of two points. We have chosen a non supersingular elliptic curve over $GF(2^n)$. Let it be $y^2+xy = x^3+a_2x^2+a_6$. Let $X_1 = (x_{10}, x_{11}, \cdots, x_{1n-1})$, $Y_1 = (y_{10}, y_{11}, \cdots, y_{1n-1})$ are the coordinates of the first point $P(X_1, Y_1)$ and $X_2 = (x_{20}, x_{21}, \cdots, x_{2n-1})$, $Y_2 = (y_{20}, y_{21}, \cdots, y_{2n-1})$ are the coordinates of the second point $Q(X_2, Y_2)$. Let (X_3, Y_3) is the resultant point $P+Q$. Here, we are giving a procedure to calculate $P+Q$.

Procedure Addition(X_1, Y_1, X_2, Y_2)

1. Begin.

2. If P is the point at infinity $\mathcal{O}$, $P+Q = Q$ and go to End.

3. Else if Q is the point at infinity $\mathcal{O}$, $P+Q = P$ and go to End.

4. Else proceed as

    - Find $X_1 + X_2$ and $Y_1 + Y_2$ using simple exclusive OR operations.
    - Find $(Y_1 + Y_2)/(X_1 + X_2)$ using one inversion and one multiplication.
    - Find $((Y_1 + Y_2)/(X_1 + X_2))^2$ using one bit rotation towards right in $(Y_1 + Y_2)/(X_1 + X_2)$.
    - Find $X_3 = ((Y_1 + Y_2)/(X_1 + X_2))^2 + (Y_1 + Y_2)/(X_1 + X_2) + X_1 + X_2 + a_2$
    - Rotate X_1 to get X_1^2.
    - Find $((Y_1 + Y_2)/(X_1 + X_2))\,(X_1 + X_3)$ using one multiplication.
    - Find $Y_3 = (Y_1 + Y_2/X_1 + X_2)\,(X_1 + X_3) + X_3 + Y_1$
    - end.

5. End.

In this way, we see that addition of two points can be performed by using one inversion and two normal basis multiplication. In the next section we will give the procedure to find double of a point.

5.3.9 Computing the Double of a Point

Let P is a point (X_1, Y_1) over the curve $y^2 + xy = x^3 + a_2x^2 + a_6$ and (X, Y) is the double of P. Here we are giving the procedure which we have implemented to find the double of a point.

Procedure Double(X_1, Y_1)

1. Begin.
2. Rotate X_1 by one bit in the right direction to get X_1^2.
3. Find $1/X_1$ using one inversion.
4. Rotate $1/X_1$ by one bit in the right direction to get $1/X_1^2$.
5. Find a_6/X_1^2 using one multiplication.
6. Find $X = X_1^2 + a_6/X_1^2$ using simple Ex-OR.
7. Find Y_1/X_1 using one multiplication.
8. Find $(X_1 + Y_1/X_1)$ using Ex-OR.
9. Find $(X_1 + Y_1/X_1)\,X$ using one multiplication.
10. Find $Y = X_1^2 + (X_1 + Y_1/X_1)\,X + X$ using Ex-OR.
11. End.

In this way, we see that the double of a point can be computed using one inversion and three normal basis multiplication.

5.3.10 Computing the Inverse Image of a Point

Let P is a point (X_1, Y_1) on the curve $y^2 + xy = x^3 + a_2x^2 + a_6$. The computation of $-P(X, Y)$ is very easy according to the following procedure.

Procedure Sub(X_1, Y_1)

1. Begin.

2. Assign X to X_1.

3. Find $Y = X_1 + Y_1$ using simple Ex-OR.

4. end.

Now we have implemented all the necessary subroutines to compute the multiple of a point P, i.e. kP. In the next section we will mention an efficient procedure to compute kP.

5.3.11 Computing the Multiple of a Point, kP

At this point, we have reached to implement the last subroutine for building an elliptic curve based cryptosystem. It is the implementation of kP. In this section, we discuss various techniques for efficient computation of kP which are useful for both hardware and software implementation.

Since this computation is equivalent to exponentiation of integers, an analog of square and multiply [Knu81] for exponentiation, named accordingly as double and add, can be used. If

$$k = \sum_{i=0}^{t-1} k_i 2^i \qquad k_i \in \{0, 1\}$$

then

$$kP = \sum_{i=0}^{t-1} k_i(2^i P)$$

Hence, we see that if k is a t bit integer then computation of kP requires $t - 1$ doublings and at the most $t - 1$ additions. The number of additions is equal to number of ones in binary expansion of k. This algorithm is very useful for hardware implementation as it does not require any precomputations or extra storage.

The addition of a point is as expensive as subtraction because inverse of a point P $(-P)$ can be computed at the cost of one addition in underlying field. Recognizing this very fact, we can improve the above algorithm by introducing a minor variation. This modified algorithm reduces the number of ones in the binary representation of k by using both subtraction and additions. In this algorithm the binary form of k is rewritten as follows. Starting from the LSB side, bits are grouped in pair of two bits (as if k is written with repect to base 4 with coefficients 0,1,2,3 in binary form). Now, whenever a pair of bits consists of two ones (coefficient 3 in base 4 representation), it is replaced by (0,-1) and 1 is added to next 2-tuple as carry. Whereas other 2-tuples, i.e. (0,0), (0,1), (1,0) are not changed. This process continues till MSB is reached. The example given below illustrates this clearly.

Example

Let $k = 98474747$. Then

$$k = \underline{01}\ \underline{01}\ \underline{11}\ \underline{01}\ \underline{11}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{01}\ \underline{11}\ \underline{11}\ \underline{10}\ \underline{11}$$

For this value of k double and add method will require 30 doublings and 21 additions. However, if we write k according to modified double and add method then we get

$$k = \underline{01}\ \underline{10} \overset{1}{\leftarrow} \underline{0\bar{1}}\ \underline{10} \overset{1}{\leftarrow} \underline{0\bar{1}}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{10}\ \underline{10}\ \underline{01}\ \underline{10} \overset{1}{\leftarrow} \underline{00} \overset{1}{\leftarrow} \underline{00} \overset{1}{\leftarrow} \underline{0\bar{1}} \overset{1}{\leftarrow} \underline{0\bar{1}}$$

Here, $\bar{1}$ represents -1 and $\overset{1}{\leftarrow}$ indicates flow of carry from right to left. Now, for the same value of k the kP can be computed with 30 doublings, 10 additions and 4 subtractions. Since, computationally subtraction and addition are same, it requires 7 less additions as compared to that required in double and add method.

If k is represented by a string of only ones, then this method shows great improvement because for $k = 2^t - 1$, simple double and method will require $t-1$ doublings and t additions, whereas modified version will require t doublings and 1 addition. In the worst case (k is represented by alternate 1 and 0's), the modified version will be same as simple binary method. Experimentally we have found that on an average total number of additions required in computation of kP are one third of number of bits in k.

Here we are giving the procedure to implement the modified double and add method for computing kP. We have sucessfully implemented this procedure.

```
Procedure ComputekP($k, P$)
1. Begin.
2. Sum = $\mathcal{O}$.
3. while ($k > 0$)
    • bitpair = $k \quad AND \quad 3$.
    • $k = k/4$.
    • If (bitpair = 3) then Sum = Sum + (-$P$) and $k = k + 1$.
    • If (bitpair = 1) then Sum = Sum + $P$.
    • $P = P + P$.
    • If (bitpair = 2) then Sum = Sum + $P$.
    • $P = P + P$.
4. End.
```

Based on the above subroutines, we have developed a software package for message encryption and decryption over $GF(2^{113})$. Simultaneously, we have implemented encryption, decryption and authentication protcols for smart card view point on the TMS320C40 digital signal processor. In the follwing sections we will dicuss about them.

5.4 A software package for elliptic curve based message encryption and decryption over $GF(2^{113})$

In some applications of public-key cryptography it is desirable, and perhaps even necessary, that the key size be as small as possible. Moreover, the cryptosystems just needs to be secure enough so that breaking it is not cost-effective.The purpose of this section is to investigate the security and practicality of elliptic curve cryptosystems with small key sizes of about 100 bits. The next two paragraphs describe some situations which come to mind where a small public key size might be desirable.

Consider the scenario where we have a small network where we would like to have secure e-mail exchange, or where we would like to have secure transmission of messages by fax. Rather than exchange public keys using certificates, key exchange is done verbally with authentication provided by voice recognition. If a symbol set consists of 32 alphanumeric characters represented by all 5-bit vectors then an n-bit key can be exchanged by representing it as an $\lceil n/5 \rceil$-symbol alphanumeric string. For n about 100 such a string is less than twice the length of most current international telephone numbers. String of this length would also be convenient for business cards, letterheads etc.

Consider also the following scenario: A software company places various programs on one distribution medium, however the purchaser can only access those programs he has paid for (the distribution medium could contain special purpose hardware that is tamper-proof for this purpose). If the user later wishes to purchase some of the other programs, he phones the company and places his request. The company in turn replies with the appropriate access information, which is digitally signed. The signature is verified by the user's terminal, and access is granted.

Most of the public-key cryptosystems are totally insecure if the key size is restricted to about 100 bits. For example, since factoring 100-bit integers can be readily done on a microcomputer, the RSA system is insecure for keys of that size. The same holds true for systems whose security is based on the intractability of the DLP in a finite field, such as the ElGamal cryptosystem. Recently La Macchia and Odlyzko [MaOd91] computed logarithms in the field $GF(p)$ where p is a 192-bit prime, while Gordon and McCurley [GoMc91] were able to compute logarithms in $GF(2^{401})$.

A good candidate that remains is the elliptic curve cryptosystem. We could sucessfully implement these systems over $GF(2^{113})$. The reason why we have chosen $n = 113$ is only that it is around 100 and optimal normal basis exist for this field. All implementations have been done in the C-language on a Pentium 100 MHz and a Pentium 150 MHz machines.

We have chosen the non-supersingular curve given as $y^2+xy = x^3+a_2x^2+a_6$, over the field $GF(2^{113})$. The computed minimal polynomial of ONBG over which the field is defined is

1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0 1 1

Here first bit is the coefficient of x^0, second of x^1 and so on. The last bit is the coefficient of x^{113}. So we can also write this polynomial as

$$1 + x + x^8 + x^9 + x^{12} + x^{13} + x^{14} + x^{64} + x^{65} + x^{72} + x^{73} + x^{76} + x^{77} + x^{78} + x^{96} + x^{97} + x^{104} + x^{105} + x^{108} + x^{109} + x^{110} + x^{112} + x^{113}$$

The curve coefficients a_2, a_6 and the publicly chosen point $P(X,Y)$ have been computed from SIMATH. The normal basis representations of these quantities are

$a_2 =$

1 1 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1
1 0 1 1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 0 0
1 1 1 0 1 1 1 1 0 1 0 0 1 1 0 0 1 0 1 0 1 1 0 1 1

$a_6 =$

1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0
0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0

The X coordinate of P is

$X =$

6-bit value	Character encoding	6-bit value	Character encoding	6-bit value	Character encoding	6-bit value	Character encoding
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

To see the encrypted message, we are giving an example of original plaintext and corresponding ciphertext for a particular encryption key and the public key of the reciever.

The plaintext is

The invention of public key crpytography by Diffe and Hellman in 1976 not only revolutionized the field of cryptography, but also had a profound effect on the direction of research in computational number theory. For the first time the question of the relative complexity of various number

theoretic tasks took on a practical urgency.

The first usable public key system introduced in 1978, was the RSA cryptosystem, which is based on the problem of factoring large integers. RSA soon became the best known and most widely used public key cryptosystem.

In 1985 a variant of discrete log cryptography was proposed, based on the DLP in the group of points of an elliptic curve defined over a finite field. Cryptosystems using discrete logarithms in this group have two potential advantages over systems based on the multiplicative group of a finite field (and also over systems based on RSA): (1) the great diversity of elliptic curves available to provide the groups; and (2) the absence of subexponential time algorithms (such as those of 'index calculus' type) that could find discrete logs in these groups.

The generated ciphertext is

ZwQc/k/HagG/rjdUHp4j81OEYIgc3enRkMv3OJVVOCRjJfUOMuRrSh4pZ/FWFDXum26xC/
ZSyH69HeE2+OzJdRiAZ3dwTevQimERzVk1ny5BKaoPo3oQlFZhW+kyMQEGolOacwicQd43
ei7TaPcEDYmo+5vKDt4ZhNtuugLuFoxnzjCdlwN5Vp09Gbm2QVFtQ0YyHlDjbOqUIea2fh
Vxwlrpteswfmk8he/hHhtP9cSXkIQ2dH8k34zL14vvvGlNB/43W0rep8unWFS88tg4oIh4
kZeZoIC69C2dGTOtcMnQJUbr3opuGlWohl4FJqPRPyXSTghAyd7dGRy8e7kUVRLENm8R5w
4GtKFinm9XYVMc5aarHL8nJJfo3f4RY7TPnOFJCkd3BP5VSuycqQnBuYJB/9kw+zgSF1rP
IozZOp/GhDzTKVZAix+Wpi/z4Vk3eUJOZwQc/k/HagG/rjdUHp4j81OEYIgc3enRkMv3OJ
VVOCRjJfUOMuRrSh4pZ/FWFDXum26xC/ZSyH6BQ2SPKtwkVF7rzUH/y1p0y3U4/utVxBak
YxLpuz7n9fLtkjfm6n+JWGkM3iWb08j3mR9I8CoXKTbu5AC2305PautehtknnQTOtx4pYC
KXMxxRLsKwIlCjEzdD2jbaKO+vR3gFHj+gCgJhmZ72ffyxEtnXPZr6ljiP/z9cTeWOmGpp
OiHrQG9miAUkF1JK/ZIPi2XRaaohwdUd3KNlSyJIngqIREP3n5q6qDaWiheUwpvTMqzKUJ
jvRB22Z4313UYl7T3c8PGLq/HibxmuSBPQ8vbIbg23/Iwv1dPUxul50UVniZUGFr63Vf0h
9/cAaOTBwoaLpCxhj9wX4ngRAdH8obePSqfoi3La/GkaLsjdmndn5Lb6ii0fX/tGVUAent
S/FFhcMIgwPRVT8r4yTnPTZ2eWYrBhqGfZlTPChtFFcVVYN2oO08/75Olm7ya5q1UxaZR9
JZlQlEBjcGf1e+a0y1SX0BaJi3DZyvejryNYpDj7Efa7SyVhbxzsuZACTf0jlARcCZkjYy

```
ACR91wWvA5Jmv50kudZIOoQZopMdggMfrFru/LEZvkMaQZej3NhcAjXB2tHhtinYsDD4a1
2HkMii8BP9RWkoV0SqD9B4r3vM2ID3RlmRFlkFOHcLr2Fzg4NKUZRWolPpMDEhBaJtGHsI
HUH+tn4+k2DHxAmJqf+ryQLeWYTrrL4ibBEtiqjnB06+TAhQGJ1Y1bqI6a71swR9u0tinW
2l9XYVMc5aarHL8nJJfo3f4RY7TPn0FJCkd3BP5N+8SN+77rRZTwP+tF9qXKv7pVhEPfLI
0KSIeGpJm8UjBCtS8TIfLuCX8qtf04ubr+l0T2vWs8gnv/o3StGDr4Nhnp0kKoYfkhw1kV
fpXU0MJBG0qZ80dTjYEyXM2ewzDVQ6IKTYYbZSem5I90jrY1pmKdX+1JoDfdZYH8z46oTo
c345GD1olgzXqZHqH7lhiIo3LY3ZM50yxcClgpNXW97H2kuTW5r5o4WXwSsRbVdXizcmZI
V23w
```

How to use this software package

This software package is very easy to handle from user point of view. An user should use it according to the following commands.

Note : Before encrypting the message, make sure that you have reciever's public key.

Encryption Procedure

1. Type your message using any editor and save it as file named "plaintxt".
2. Now give the command "ecrypt" at your DOS prompt. It will demand for encryption key. Enter your chosen encryption (private) key. Now wait for some time for encryption to be completed.
3. After completion of encryption, open the file "ciphertx" to see the transmitted ciphertext.

Decryption Procedure

1. Type the command "dcrypt" at your DOS prompt. It will demand for your decryption key. Enter the decryption key and wait for some time for the decryption to be completed.
2. After completion of decryption, the user should open the file "rplaintx" to read the recieved message which is the original plaintext.

Performance

With a particular value of encryption and decryption key, the 106 KB file could be encrypted in 880 seconds with the encryption rate of 124 bits/sec. The encrypted file could be decrypted in 640 seconds with the decryption rate of 170 bits/sec. The code occupies about 0.5 MB memory.

5.5 Implementation of Encryption/Decryption over TMS320C40 Digital Signal Processor Using Optimizing C Cross Compiler

Work that motivated ours is Barrett's, Wiener's and Davio *et al's* [PB86, DDFG83]. Barrett observed the effectiveness of digital signal processors for cryptography and presented an implementation of RSA on Texas Instruments' TMS32010 [PB86]. Davio *et al* made considerable progress in efficient TECHNIQues for DES [DDFG83]. Here we have implemented the elliptic curve public key cryptosystems on the Texas Instrument's TMS320C40 25 MHz digital signal processor.

The approach we adopted is based on optimizing C cross compiler. This compiler takes the ANSI C programs as input and generates the TMS320C40 assembly language programs as output. For detailed information on hardware architecture and C cross compiler, the readers may refer to TMS320C4x User's Guide and TMS320 Floating-Point DSP Optimizing C Compiler User's Guide.

For this implementation, we are provided with the dsp board installed within a 40 MHz Intel486 PC. The dsp board is the target processor while the PC is the host processor. For every operation to be implemented, we have to write two prgrams. One is for the dsp board (simply the ANSI C source code of an operation) and other one is for the host to communicate with the dsp board. The function of the host program is to take input from the keyboard or give output to the screen. Simultaneouly it downloads the data to the board and after execution done by the board it uploads the reults from the board. For writing the host program, we are provided with a Run-Time Library Reference containing the uploading and downloading routines. For more information see SPIRIT-40 ISA User's Guide.

We could sucessfully do the implemetation of field operations and elliptic curve operations and then message encryption and decryption based on these over much higher fields (up to $GF(2^{400})$). For examples we have taken $GF(2^{135})$, $GF(2^{155})$, $GF(2^{303})$. The curve chosen is non-supersingular, i.e. $y^2 + xy = x^3 + a_2x^2 + a_6$.

Over $GF(2^{135})$

The generated minimal polynomial of ONBG, over which the field is defined, is

1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1

The curve coefficients and publicly chosen point P are:

$a_2 =$

1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1

$a_6 =$

0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1

The publicly chosen point $P(X, Y)$ is

$X =$

1 0 0 0 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1
0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0 1 0 C 1 1 0 1 0 0 0 0 0 1 1 1 0 1 0 0 0
1 0 0 0 0 0 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0

$Y =$

0 1 0 1 1 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0
0 0 0 1 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 0
0 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 1 1 0 0 1 1 0 0

Over $GF(2^{155})$

The generated minimal polynomial of ONBG is

1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0
1 0 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1
1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 1 1

$a_2 =$

1 1 0 1 1 1 1 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1
1 0 0 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 1 0 0 1 1 0 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0
0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1
1 1 0 1 0 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 1

$a_6 =$

0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1
1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1
1 1 1 0 1 0 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 1 0 1 1 0
1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1

The point $P(X,Y)$ is

$X =$

1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0 0 1 0 0 0 0 1 0
0 1 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 1 0 0
0 0 1 1 0 1 0 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 1 0 1 1 1 0 1 0 0 1 1 1 1 0 0 0 1 1 1 0 0
1 1 0 1 1 0 1 1 1 1 0 0 1 1 1 1 0 1 1 0 1 0 0

$Y =$

0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 1 0 0 0 0 0
0 1 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 1 0
1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0 1
0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1

Over $GF(2^{303})$

The generated minimal polynomial of ONBG is

1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0
0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1

$a_2 =$

0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0 0 0 1 1
1 0 0 1 0 1 0 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 0 0 1 1
0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1
1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0
1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0

001101111111101011101100001011110110101101011111

$a_6 =$

1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1
1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 0
1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0 0 0
0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1
1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1
111101111101111101111101111111111111111111111111

The Point $P(X, Y)$ is

$X =$

1 0 1 1 0 1 1 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1 1
0 1 1 0 1 0 1 0 1 0 1 1 0 1 1 1 1 1 0 0 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 1 1 1
1 0 1 0 0 1 1 1 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 1 0 1 1 1 0 0 0 1 1 1 0 1 0 0 0
0 1 1 1 1 1 0 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 0 1 1 1 0 1 0 1 1 1 1
1 0 1 1 0 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 0
0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 0 1 1 1 1 0 1 1 0 1 0 1 1 1 0 0 1 1 0 0 0
000101100110110101000101110100011100011111111011

$Y =$

0 0 1 1 0 0 1 1 1 1 0 1 0 1 0 1 1 0 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0
1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 0 1 1 0 1 1 1 0 1
1 0 1 1 0 1 0 0 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 1 1 1
0 0 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 1 0 0 0 0 1 0 0 1
1 1 1 1 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1
1 1 1 0 0 0 1 1 1 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0
0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 1 1

Here all the curve coefficients and point's coordinates are given in normal basis representation. First bit denotes LSB and the last bit denotes MSB.

The following tables show the timing results which we have obtained for various field size.

	$GF(2^{113})$	$GF(2^{135})$	$GF(2^{155})$	$GF(2^{191})$	$GF(2^{303})$
one field multiplication	3 sec.	5 sec.	7 sec.	13 sec.	53 sec.
one field inversion	22 sec.	44 sec.	71 sec.	160 sec.	638 sec.

Table 3. Times for field operations on TMS320C40 25MHz.

	$GF(2^{113})$	$GF(2^{135})$	$GF(2^{155})$	$GF(2^{191})$	$GF(2^{303})$
one curve addition	29 sec.	54 sec.	86 sec.	187 sec.	745 sec.
one curve doubling	32 sec.	58 sec.	93 sec.	200 sec.	798 sec.

Table 4. Times for elliptic curve operations on TMS320C40 25 MHZ.

With these timings results we conclude this chapter. For our implementation work, we do not claim it to be efficient from memory and speed point of view. More efficient codes may be written using other programming techniques and algorithm modifications. But we have tried our best throughout whole implementation work.

Chapter 6

Conclusions and Future Work

After discussing various aspects involved in the implementation of efficient and secure elliptic curve cryptosystems, we conclude the thesis with a review of main points and scope for future work.

6.1 Conclusions

The aim of this thesis was mainly concentrated on the software implementation of efficient and secure elliptic curve public key cryptosystems over Galois field $GF(2^n)$. while writing the thesis, this aim was kept in the mind so that we covered only those issues of elliptic curves which are necessary and sufficient for the implementation point of view. We designed an elliptic curve based smart card cryptosystems. Elliptic curves were found to be very effective and advantageous as compared to RSA for smart card design. Since the arithmetic in Galois field is time consuming, their efficient implementation was our main issue. We looked the effectiveness of optimal normal bases for the implementation to be efficient. We gave efficient procedures in implementable form for Galois field operations as well as for elliptic curve operations. The effectiveness of these procedures were tested through implementation and checked both on Pentium and digital signal processor. A software package could be developed for message encryption and decryption over $GF(2^{113})$ for e-mail and fax security point of view and successfully tested on Pentium processors installed with DOS and Linux platforms. Encryption and decryption over the fields as large as $GF(2^{400})$ could be successfully done on the TMS320C40 25 MHz digital signal processor. Elliptic curve based ElGamal scheme over $GF(2^n)$ was employed for message encryption and decryption. Even though the encryption rate is too low, the software implementation can be used to study the implementation over various curves. An important issue about implementation is that all the work has been done using ANSI C language without using any ready made package such as SIMATH. For applications such as e-mail security, smart card programming etc., this approach is very useful because installing the whole package is not possible due to limited memory space.

6.2 Future Work

In this thesis, we implemented the software for elliptic curve based data encryption and decryption, particularly for smart card based applications. In future, this work can be extended to implement a whole smart card based off-line electronic payment system used for remote shopping over the Internet. Internet Commerce is the hottest topic of today and of the future. Most of the presently used electronic payment systems are credit card based, on-line payment systems employing RSA, DSS public key algorithms. These algorithms can be replaced by elliptic curve public key algorithms in future. The disadvantage of on-line payment system is that a couple of banks have to be involved during the whole transaction causing wastage of communication bandwidth and time. An alternative approach is the smart card based off-line payment system in which the card plays the role of banks and does itself all the authentications, encryption and decryption required during transaction. The card does so by using the software written into its EEPROM. The elliptic curve based software may be very useful for such type of applications.

Appendix A

An Introduction to Electronic Commerce

A.1 Introduction

In this age of networking and inter-networking, business is moving from face-to-face trading, mail order, and telephone order to electronic commerce over open networks such as the Internet. If you have an PC, a credit card and Internet access of course, you can purchase goods remotely without need to go to the shopkeeper. If a smart card reader is attached to your workstation, you can make your business transaction by just inserting the smart card into the reader. Since Internet is an insecure publicly available open network, securing payments over it connecting commercial severs and consumer work-stations poses challenges of a new dimension. In the following section, we attempt to provide an overview of electronic payment systems focusing on issues related to their security.

A.2 Electronic Payment Models

In a traditional payment system, the participants involved are payer, payee and and at least one financial institution (usually bank) to guarantee the validity of money. The same involve in the electronic payment system. Only, the role of financial institution is divided into two parts: an issuer (used by the payer) and an acquirer (used by the payee).

How the real money reaches to the payee. The flow of real money from payer to payee is made possible via payer to issuer, issuer to acquirer, acquirer to payee. According to the payment method,the payment models can be categorized as

prepaid cash-like payment system

A certain amount of money is taken away from the payer (for example, by debiting that amount from the payer's bank account) before purchase are made. This amount of money can be used for payments later. Smart card-based electronic purses, electronic cash as well as bank cheques fall in this category. The model is shown as below.

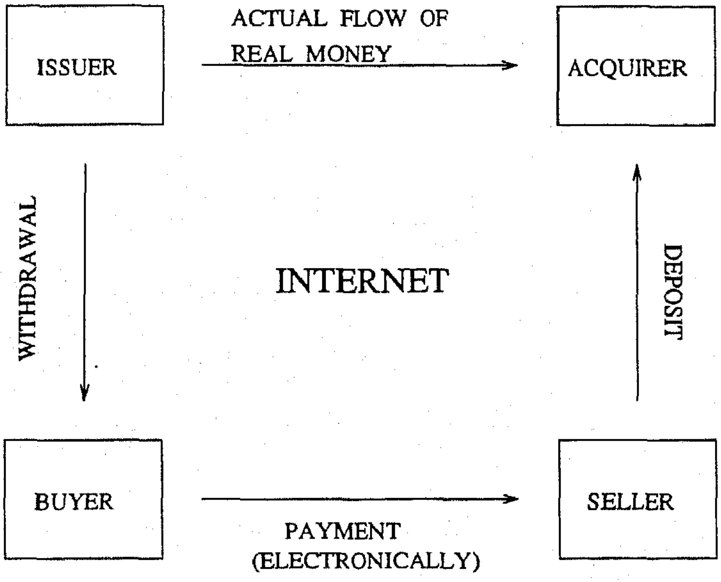

Figure A.1: Cash-like Payment System

Pay-now payment Systems

In pay-now payment systems, the payer's account is debited at the time of payment. ATM card based systems fall into this category.

Pay-later (credit) payment systems

In these payment systems, the payee's bank account is credited the amount of sale before the payer's account is debited. Credit card systems fall into this category. Typical flows of these systems are shown below. As a payment is always done by sending some sort of "form" from payer to payee (cheque, credit card slip, etc.) we call these systems cheque-like.

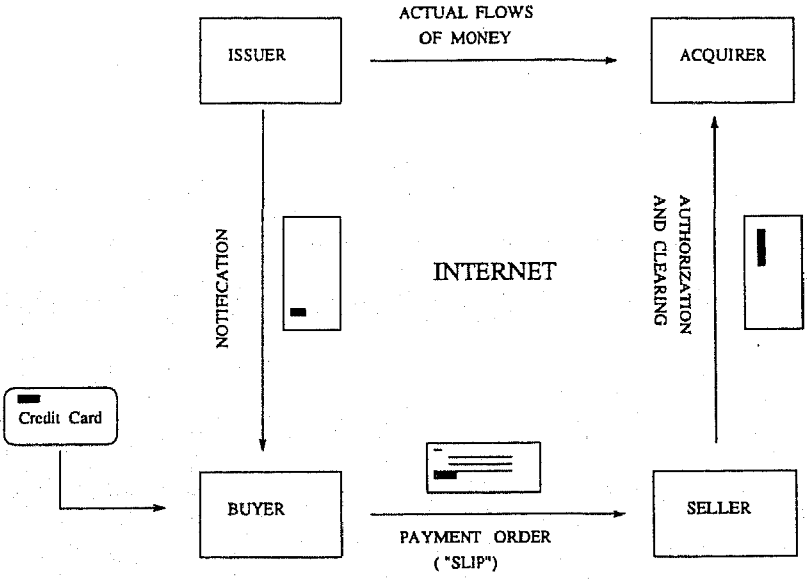

Figure A.2: Cheque-like Payment System

A.3 Security Requirements

In general, one or more of the following security requirements must be met withir an electronic payment system.

Integrity and Authorization

No money is taken from a user unless a payment is explicitly authorized by him. Moreover users might require not to receive any payment without their explicit consent; this is desirable when users wants to avoid unsolicited bribery.

Confidentiality

The knowledge about various pieces of information related to a transaction such as identity of payer/payee, purchase content, amount, etc. must be restricted only to the participants involved.

Availability and Reliability

All parties require the ability to make or receive payments whenever necessary. Payment transactions must be atomic: they occur entirely or not at all, but never hang in an unknown or inconsistent state. No payer would accept a loss of money due to a network or system crash.

A.4 On-line vs. Off-line Payment Systems

In on-line payment systems, an authorization server (usually as part of the issuer or acquirer) has to be involved in each payment. While in off-line systems, only payer and payee come in the picture without involving any third party during payment.

The obvious problem with off-line payments is how to prevent payers from spending more money than they actually possess. Since e-money is just a bunch of bits, a piece of e-money is very easy to duplicate. A trivial e-money system would allow to copy of a piece of e-money and spend both copies. And the user could become a millionaire in a matter of a few minutes. This problem is known as **double spending problem.**

On-line systems prevent double spending problem by requiring seller to contact the bank's computer with every sale. The bank computer maintains a database of all the spent pieces of e-money and can easily indicate to the seller if a given piece of e-money is still spendable.

Off-line systems detect double spending by using tamper resistant hardware, such as **smart cards**, at the payer end. The chip inside the smart card keeps a mini database of all the pieces of e-money spent by that smart card. If the owner of the smart card attempts to copy some e-money and spend it twice, the embedded chip would detect the attempt and would not allow the transaction.

On-line payment systems obviously require more communication, but not necessarily tamper-resistant hardware. In general, they are considered more secure than off-line systems.

A.5 Anonymity of Payer

Some payment systems provide payer anonymity and untraceability. Payers prefer to keep their everyday payment activities private. Certainly they do not want unrelated third parties to observe and track their payments. Often, they prefer the payees (shops, publishers, etc.) and in some cases even banks to be incapable of observing and tracking their payments.

Whereas **anonymity** simply means that the payer's identity is not used in payments, **untraceability** means that, in addition, two different payments by the same payer cannot be linked. By encrypting all flows between payer and payee, all payment systems could be made untraceable by outsiders. Payer anonymity with respect to the payee can be achieved by using pseudonyms instead of real identities.

A.6 Some Proposed Electronic Payment Systems

Secure Electronic Transactions (SET)

This is likely to be widely adopted for credit card payments over the Internet. SET concentrates on securely communicating credit card numbers between a payer and an acquirer. In our classification, SET falls under the "cheque-like" model. In a first handshake the seller authenticates itself to the payer and all offer and payment data are fixed. The payer then generates a payment slip using a sophisticated encryption scheme which protects the sensitive payment information (e.g., credit card number), limits the encryption to selected fields to ease export approval, cryptographically ties the order information, and minimizes exposures of the user's privacy. This slip is then signed by the payer to authorize the payment and is sent to the seller, who sends it to its acquirer to authorize and capture the payment. The acquirer checks all signatures and the slip, verifies over the existing network the creditability of the payer and sends - depending on the outcome of this operation - either a positive or negative signed acknowledgment back to seller and buyer. For getting technical information about SET, refer [SET].

E-Cash

E-Cash, a cash-like payment system, provides high levels of anonymity and untraceability. E-Cash is based on the concept of **blind signatures.**. When an entity A wants to obtain a blind signature on a message m fro an entity B, A generates a blinded message m' from m and requests B to sign m' and return the blind signature on m, $sign_B(m')$, to A. The blinding transformation is such that:

- B (and no one else) can construct $sign_B(x)$ given x but anyone can verify it.
- A (and no one else) can derive the signature on m (i.e., $sign_B(m)$) given the blind signature on it, $sign_B(m')$.

To know more about these systems, refer [Ecash].

With these proposed systems, we conclude this brief introduction on electronic payment systems. A lot of information is available on the Internet on this subject. Interested may refer to [SEMPER].

Bibliography

[AMV93] G. Agnew, R. Mullin, and S. Vanstone. An Implementation of elliptic curve cryptosystems over F_{2^m}. *IEEE Journal on selected areas in Communications,* June 1993.

[Apo76] Tom M. Apostol. *Modular Functions and Dirichlet Series in Number Theory.* Number GTM-41. Springer-Verlag, 1976.

[BJN94] P.B. Bhattacharya, S.K. Jain, and S.R. Nagpaul. *Basic Abstract Algebra.* Cambridge University Press, 1994.

[BM91] J. Buchmann and V. Muller. *Computing the number of points of elliptic curves over finite fields*, presented at International Symposium on Symbolic and Algebric Computation, Bonn, July 1991.

[Bri89] E.F. Brickell. A Survey of Hardware Implementation of RSA. *LNCS: Advances in Cryptology-CRYPTO'89*, 1989.

[Cass91] J. Cassels, *Lectures on Elliptic Curves,* Cambridge University Press, 1991.

[Cha88] J.S. Chahal. *Topics in Number Theory.* Plenum Press, 1988.

[Coh78] Harvey Cohn. *A Classical Invitation to Algebraic Numbers and Class Fields.* Springer-Verlag, 1978.

[DDFG83] M. Davio, Y. Desmedt, M. Fosseprez, R. Govaerts, J. Hulsbosch, P. Neutjens, P. Piret, J.J. Quisquater, J. Vandewalle and P. Wouters. Analytical Characteristics of the DES. *Advances in Cryptology: CRYPTO'83*, 1984.

[Denn] Dorothy Elizabeth Robling Denning *Cryptography and Data Security*, Purdue University.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory,* November 1976.

[DH79] Whitfield Diffie and Martin E. Hellman. Privacy and Authentication: An Introduction to Cryptography. *Proceding of the IEEE,* March 1979.

[DJ91] Stephen R. Dusse and Burton S. Kaliski Jr. A Cryptographic Library for the Motorola DSP56000. *LNCS:Advances in Cryptology-EUROCRYPT'90,* 1991.

[DVJ96] Jean Francois Dhem, Daniel Veithen. and J.J. Quisquater. SCALP:Smart Cards For Limited Payment Systems. *IEEE Micro,* June 1996.

[Ecash] DigiCash public documentation at URL : *http://www.digicash.com*

[ElG85] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory,* 1985.

[FOM92] Atsushi Fujioka, Tatsuaki Okamato, and Shoji Miyaguchi. ESIGN: An Efficient Digital Signature Implementation for Smart Cards. *LNCS:Advances in Cryptology'92*, 1992.

[Fra95] John B. Fraleigh. *A First Course in Abstract Algebra, Narosa Publishing House*, 1995.

[Fult69] William Fulton. Algebraic Curves. W.A. Benjamin, Inc, 1969.

[Gemp] Gemplus' Home Page. *http://www.gemplus.com*

[GoMc91] D. Gordon and K. McCurely. Computation of Discrete Logarithms in $GF(2^n)$, *CRYPTO'91*, 1991.

[GoMc92] D. Gordon and K. McCurely. Massively Parallel Computation of Discrete Logarithms. *CRYPTO'92*, 1992.

[GuUg86] L.C. Guillou and M. Ugon. Smart Card: A highly reliable and portable security device. *LNCS:Advances in Cryptology, CRYPTO'86*, 1987.

[Hast85] J. Hastad. On using RSA with Low Exponent in a Public Key Network. *Crypto'85*, 1985.

[HaWa88] M.E. Haykin and R.B. Warnar. *Snart Card Technology, New Methods for Computer Access Control,* NIST, Sept. 1988.

[Hec93] Erich Hecke. *Lectures in Theory of Algebraic Numbers.* Number GTM-77. Springer-Verlag, 1993.

[HMV92] G. Harper, A. Menezes and S. Vanstone. Public Key Cryptosystems with very small key length, *Advances in Cryptology, EUROCRYPT'92.*

[Hus87] D. Husemoller. *Elliptic Curves.* Springer-verlag, New York, 1987.

[IR82] Kenneth Ireland and Michael Rosen. *A Classical Introduction to Modern Number Theory.* Number GTM-84. Springer-Verlag, 1982.

[ITT86] T. Itoh, O. Teechai and S. Tsujii. A fast algorithm for computing multiplicative inverses in $GF(2^n)$ using normal bases. *J. Society for Electronic Communications*, 1986.

[IWSD92] P. Ivey, S. Walker, J. Stern and S. Davidson. An ultra-high speed public key encryption processor. *Proceedings of IEEE Custom Integrated Circuits Conference*, Boston, 1992.

[KK94] H. Kuwakado and K. Koyama. *Security of RSA-type cryptosystems over elliptic curves against Hastad attack*, Electronics Letters, Vol. 30, No. 2, 27th October, 1994.

[Knu81] D.E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms.* Number MA. Addison-Wesley, Reading, 1981.

[Kob84] N. Koblitz. *Introduction to Elliptic Curves and Modular Forms*, Springer-Verlag, New York, 1984.

[Kob87] N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 1987.

[Kon91] Hans Peter Konigs. Cryptographic Identification Methods for Smart Cards in the Process of Standardization. *IEEE Communication Magazine*, June 1991.

[KOT94] K. Kurosawa, K. Okada, S. Tsujii. Low Exponent Attack against Elliptic Curve RSA, *ASIACRYPT'94, LNCS, Vol. 917.*

[Kumar96] S. Kumar. Implementation of Public Key Elliptic Curve Cryptosystems, *M.Tech. Thesis, Deptt. of Electrical Engg.*, IIT Kanpur, 1996.

[Lang78] S. Lang. Elliptic Curves: *Diophantine Analysis,* , Springer-Verlag, 1978.

[LdNd] R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications*, Cambridge University Press.

[MaOd91] B. La Macchia and A. Odlyzko. Computation of discrete logarithms in prime fields, *Designs, Codes and Cryptography*, 1991.

[McEl] Mc Elice. *Finite Fields for Computer Scientists and Engineers.* Kluwer Academic Publications, 1992.

[Men93a] Alfred J. Menezes, editor. *Applications of Finite Fields.* Kluwer Academic Publishers, 1993.

[Men93b] Alfred J. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993

[Mill85] V. Miller. Uses of elliptic curves in cryptography. *Advances in Cryptology-CRYPTO'85, LNCS*, 1986.

[Miy91] Atsuko Miyaji. On Ordinary Elliptic Curve Cryptosystems. *LNCS: Advances in cryptology, ASIACRYPT'91*, 1991.

[Miy92] Atsuko Miyaji. Elliptic Curves over F_p Suitable for Cryptosystems. *LNCS:Advances in cryptology: AUSCRYPT'92*, 1992.

[More91] C. Moreno. *Algebraic Curves over Finite Fields*, Cambridge University Press, 1991.

[MOVW88] R. Mullin, I. Onyszchuk, S. Vanstone and R. Wilson. Optimal Normal Bases in $GF(p^n)$. *Discrete Applied Mathematics*, 1988/89.

[MV90] A. Menezes and S. Vanstone. The Implementation of Elliptic Curve Cryptosystems, *AUSCRYPT'90, LNCS, Vol. 453*

[MV93] A. Menezes and S. Vanstone. Elliptic Curve Cryptosystems and their implementation. *Journal of Cryptology*, 1993.

[NBS77] National Bureau of Standards, Data Encryption Standard, *Federal Information Processing Standard*, U.s. Department of Commerce, FIPS PUB 46, Washington, DC, 1977.

[NM95] David Naccache and David M'Raihi. Cryptographic Smart Cards. *IEEE Micro*, June 1995.

[Ono90] Takashi Ono. *A intoduction to Algebraic Number Theory.* May 1990.

[OSA92] Holger Orup, Erik Svendsen, and Erik Andreasen. VICTOR: An Efficient RSA Hardware Implementation. *LNCS: Advances in Cryptology'92*, 1992.

[Pank97] Pankaj Bansal. *Construction of Elliptic Curves for Efficient and Secure Cryptosystems*, M.Tech. Thesis, Deptt. of EE, IIT Kanpur.

[PB86] Paul Barrett. Implementing the RSA public key encryption algorithm on a standard digital signal processor. *LNCS:Advances in Cryptology, CRYPTO'86*, 1987.

[PS92] A. N. Parshin and I.R. Shafervich. *Number Theory II: Algebraic Number Theory.* Number EMS-60. Springer-Verlag, 1992.

[RSA78] R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, 1978.

[ROS94] H.E. Rose. *A course in Number Theory.* Clarendon Prss, 1994.

[Sch85] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p, *Mathematics of Computation*, 1985.

[Sch89] C.P. Schnorr. Efficient Identification and Signatures for Smart Cards. *LNCS:Advances in Cryptology-CRYPTO'89*, 1989.

[Sch93] Bruce Schnneier. *Applied Cryptography.* John Wiely and Sons, Inc., 1993.

[Seg80] Sanford L. Segal. *Nine Introduction in Complex Analysis.* Number North Holland Mathematics Studies-53. North Holland Publishing Company, 1980.

[SEMPER] URL: *http://www.semper.org/sirene/outsideworld/ecommerce.html*, A Study on Secure Electronic Market Place for Europe.

[SET] URL: *http : //www.mastercard.com/set/set_bk2.pdf*, A technical specifaction on SET protocol.

[Sil85] Joseph H. Silverman. *The Arithmatic of Elliptic Cutves.* Number GTM-106. Springer-Verlag, 1985.

[Sil94] Joseph H. Silverman. *Advanced Topics in the Arithmatic of Elliptic Curves.* Number GTM-151. Springer-Verlag, 1994.

[Sim91] G. simmons, editor. *Contemporary Cryptography: The Science of Information Integrity.* IEEE Press, New York, 1991.

[Sta95] William Stalling. *Network and Internetwork Security Principles and Practice*, IEEE Press, Prentice Hall, Englewood Cliffs, New Jersy 07632, 1995.

[VVDJ92] Andre Vandemeulebroecke, Etienne Vanzieleghem, Tony Denayer, and Paul G. A. Jespers. A single Chip 1024 bits RSA processor. *LNCS: Advances in Cryptology'92*, 1992.

[WQ90] Dominique de Waleffe and Jean Jacques Quisquater. CORSAIR:A smart card for public key cryptosystems. *LNCS:Advances in Cryptology, CRYPTO'90.*